Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Sebastien Strban

Vizualizacija trkov delcev v hadronskih trkalnikih

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matija Marolt
Somentor: as. dr. Ciril Bohak

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu preučite področje vizualizacije fizikalnih eksperimentov, predvsem trkov delcev v fiziki visokih energij. Izdelajte vizualizacijo tovrstnih eksperimentov, pri čemer naj bo vizualizacija čimbolj interaktivna in široko dostopna. Posvetite se predvsem optimizaciji izrisa in prenosa podatkov na grafično kartico. Sistem ovrednotite.

Sprva bi se rad zahvalil članom Laboratorija za računalniško grafiko in multimedije za podporo in omogočanje mojega dela v okolju laboratorija. Zahvaljujem se mentorju izr. prof. dr. Matiji Maroltu za podporo med delom in ker mi je tekom študija vzbudil zanimanje za to delo. Zahvaljujem se somentorju as. dr. Cirilu Bohaku, ki me je med delom vodil, usmerjal in podajal svoje mnenje. Obema se zahvaljujem tudi za podporo med pisanjem diplomskega dela.

Za sodelovanje se zahvaljujem članom skupine za vizualizacijo pri organizaciji CERN in posameznih fizikalnih eksperimentih (ATLAS, CMS in Alice).

Za podporo pa se zahvaljujem svoji družini in prijateljem. Brez njih bi mi le težko uspelo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
2D	two-dimensional	dvodimenzionalno
3D	three-dimensional	tridimenzionalno
API	application programming interface	aplikacijski programski vmesnik
CPE	central processing unit	centralno procesna enota
CRT	cathode ray tube	katodna cev
GPE	graphics processing unit	grafično procesna enota
HEP	high energy physics	fizika visokih energij
LHC	Large Hadron Collider	Veliki hadronski trkalnik
VR	virtual reality	navidezna resiničnost

Povzetek

Naslov: Vizualizacija trkov delcev v hadronskih trkalnikih

Avtor: Sebastien Strban

V delu predstavljamo sistem za vizualizacijo trkov osnovnih delcev, v fiziki visokih energij, ki smo ga razvili s sodelovanjem s CERN-om. Pri eksperimentih v fiziki osnovnih delcev raziskovalci zajamejo ogromno količino podatkov, na podlagi katerih poskušajo rekonstruirati dogajanje. Za boljše razumevanje dogajanja je potrebna ustrezna vizualizacija, kar v primeru velike količine podatkov predstavlja svojevrstni izziv. Še posebej v primeru, ko želimo takšno količino podatkov vizualizirati v realnem času in z visoko stopnjo interaktivnosti.

Zaradi želje po čim širši dostopnosti smo se odločili za implementacijo v spletnih tehnologijah. V ta namen smo razvili sistem za vizualizacijo opisanih eksperimentov v spletnem brskalniku z uporabo tehnologije WebGL. Sistem je optimiziran za čim hitrejše in učinkovitejše delovanje na grafičnih procesorskih enotah. Pri tem smo se osredotočili na minimizacijo klicev za izris, in minimizacijo prenosa podatkov na grafično kartico. To se je izkazalo za zelo učinkovito rešitev, kar potrjuje performančna analiza razvitega sistema.

Sistem smo testirali tudi z uporabo ogrodja Med3D, ki uporablja hibridni način upodabljanja. Ta združuje upodabljanje preko brskalnika z upodabljanjem na namenskem strežniku.

Ključne besede: vizualizacija trkov delcev, CERN, WebGL, klic za izris, Med3D.

Abstract

Title: Visualization of particle collisions in hadron colliders

Author: Sebastien Strban

In this work we present a visualization system for particle collisions in high-energy physics, which we developed in cooperation with CERN. During the experiments in particle physics a large amount of data is acquired, used for event reconstruction. For better understanding suitable visualization is needed. This poses a big challenge due to large amount of data, especially when we want to achieve real-time interactive functionality.

For wider accessibility we decided to implement the visualization in web-based technologies. We developed a visualization system using WebGL, optimized for fast and efficient use with GPUs. We focused on minimization of draw calls and minimization of data transfer to the the GPU. This proofs as an efficient solution confirmed by performance analysis of the developed system.

The system was also tested using the Med3D system, which uses a hybrid rendering mode. This combines rendering through a browser with rendering on a dedicated server.

Keywords: visualization of particle collisions, CERN, WebGL, draw call, Med3D.

Poglavje 1

Uvod

Vizualizacija predstavlja zelo močno in pomembno orodje. Uporaba vizualizacije je velikokrat samoumevna, a ima zelo veliko vrednost. Človeško oko in vid sta ena najpomembnejših čutil in procesov, ki se dogajajo na zaznavnem nivoju, vizualizacija pa lahko ogromno doprinese k razumevanju stvari, ki nam s prostim očesom niso vidne.

Vizualizacijo podatkov srečamo na področju znanosti, multimedije, umetnosti, grafičnega oblikovanja in drugod. Primerna vizualizacija nam omogoča boljše preučevanje in razumevanje obravnavanih podatkov in informacij. Omogoča nam kontroliran in obvladljiv pogled na opazovani sistem, ki je lahko v realnosti preobsežen in preveč kompleksen zaradi fizičnih in psiholoških omejitev.

Vizualizacijo pogostokrat uporabimo pri prikazovanju velike količine podatkov in informacij, ki so v surovi obliki lahko nejasni in neobvladljivi ali pa zgolj preobsežni. S primerno izbrano vizualizacijo rešujemo take probleme z namenom jasnosti in prikaza objektivne realnosti.

Vizualizacija predstavlja zelo pomembno vlogo na vseh področjih znanosti. S primerno vizualizacijo omogočimo lažje razumevanje obravnavanega problema, spremljamo obnašanje določenega sistema, ali morda celo opazimo določene lastnosti, ki iz surovih podatkov niso opazne. Vizualizacija je zelo pomembna na področju fizike, natančneje pri opazovanju dogajanja

v kvantni mehaniki, ki lahko hitro postane kontraintuitivna. Primerna vizualizacija omogoči pravilnejše, natančneje in boljše razumevanje delovanja tovrstnih sistemov in izpopolnjevanje znanstvene teorije.

1.1 Motivacija

Z napredkom v tehnologiji je vizualizacija postala zelo pomembna pri trkih osnovnih delcev, ki se uporablja za temeljne raziskave v fiziki delcev. Tovrstni eksperimenti so nadzorovani z uporabo množice različnih detektorjev, prilagojenih za zaznavanje izbranih tipov delcev. Pri trkih delcev z visokimi energijami pride do razpada v še bolj osnovne delce, ki pa jih želimo zaznati in zaradi boljšega razumevanja tudi vizualizirati.

V fiziki so subatomski delci definirani kot delci, ki so veliko manjši od atomov. Subatomske delce lahko delimo na elementarne in sestavljene delce. Elementarni delci niso izdelani iz drugih delcev, medtem ko so sestavljeni zgrajeni iz elementarnih delcev. Trenutno v znanosti za elementarne delce štejejo fermioni (kvarki, leptoni, antiquarki in antileptoni) in bozoni. Fermioni so delci, ki sestavljajo materijo in antimaterijo. Bozoni so povzročitelji sile interakcije med fermioni.

Velik premislek je sprožila ugotovitev, da se lahko svetloba obnaša kot delec (fotoni) in kot valovanje. To je privedlo do novega koncepta, ki ga imenujemo dualnost delec-valovanje. Odraža, da se kvantni delci obnašajo tako kot delci in valovi. Z novim premislekom se je pojavil tudi koncept principa nedoločenosti. Ta opredeljuje, da se nekaterih lastnosti delca, kot so položaj in gibalna količina, ni mogoče povsem natančno sočasno izmeriti. Ugotovljeno je bilo tudi, da dualnost delec-valovanje ne velja le za fotone in osnovne delce, temveč tudi za vse večje in masivnejše delce.

Eden izmed pristopov preučevanja delcev je z uporabno pospeševalnikov delcev. Pospeševalniki se uporabljajo za temeljne raziskave v fiziki delcev. Trenutno najzmogljivejši pospeševalnik je Veliki hadronski trkalnik (angl. Large Hadron Collider - LHC) v bližini Ženeve v Švici. Veliki hadronski

trkalnik je zgradila Evropska organizacija za jedrske raziskave (CERN). Nekateri drugi pospeševalniki so npr. KEKB v KEK (Japonska) in RHIC v nacionalnem laboratoriju Brookhaven (ZDA). Veliki hadronski trkalnik je pospeševalnik, ki lahko pospeši dva žarka protonov na energijo 6,5 TeV (tera elektron voltov) in povzroči čelno trčenje tako, da ustvarja energijo središča mase 13 TeV.

Proces eksperimenta zahteva pospeševalnike, močne elektromagnete in detektor, ki je zgrajen iz plasti kompleksnih poddetektorjev. Posamezni dogodek v CERN-u se pripravi s pospeševanjem delcev (tesno razmaknjeni skupki) do visokih energij, preden jih nadzorovano preusmerijo v trk.

To storijo z uporabo pospeševalnika, ki z vplivom električnih polj povečuje hitrost in energijo snopa (angl. beam) delcev in magnetna polja nadzoruje smer in fokus. Trk skupine delcev povzroči razpad na množico osnovnejših delcev. Delci, ki nastanejo ob trkih, običajno potujejo po ravnih črtah, a s prisotnostjo magnetnega polja, ki ga ustvarijo elektromagneti okoli poddetektorjev, njihove poti postanejo ukrivljene. Tisti delci, ki nosijo naboj, se po trku pod vplivom močnega magnetnega polja odklonijo in zaidejo skozi množico različnih poddetektorjev. Poddetektorji so prilagojeni na zaznavanje določenega tipa delca ali pa določene lastnosti delca. Tam delci interaktirajo s poddetektorji, kjer se izmerijo njihove lastnosti, kot so položaj, hitrost, masa, naboj, gibalna količina in energija. Na podlagi teh informacij se delcem določi tudi njihova identiteta.

Takšen dogodek je v eksperimentu predstavljen z množico točk v prostoru, kjer so bili delci zaznani in izmerjene njihove lastnosti. Za nadaljnje študije je potrebno rekonstruirati poti vseh posameznih nastalih delcev, kar predstavlja zelo težak problem. Da pri rekonstrukciji lažje opredelimo pravilnost posamezne poti in morebitne napake (npr. zaradi prisotnosti šuma), potrebujemo primerno vizualizacijo.

1.2 Pregled področja

Vizualizacija podatkov se je začela že s samim začetkom znanosti računalništva in informatike. V zgodnjem obdobju se je pojavila kot risalnik grafov na neskončen papir, ali pa kot enostavni prikazovalniki CRT, ki so vsebino izrisovali s katodno cevjo (z elektromagnetnim fokusiranjem in odklonom). Temu je sledila črtna grafika, kmalu za tem pa se je že pojavila rastrska grafika.

Razvoj namenske strojne opreme je omogočil vedno večjo zmogljivost in posledično tudi obdelavo vedno večje količine podatkov in stopnje realizma v vizualizacijah. Razvoj nas je privedel do grafičnih pospeševalnikov z ločenim procesorjem in spominom, ki so prilagojeni prikazovanju računalniške grafike. Grafični pospeševalnik nam omogoča visoko stopnjo paralelizma, s pomočjo katerega lahko učinkovito, hitro in interaktivno, v realnem času, obdelujemo in upodabljamo podatke.

Razvita je bila tudi programska podpora vizualizaciji, kot je OpenGL [19]. To je specifikacija standarda, ki določa programski vmesnik za pisanje računalniških programov, ki prikazujejo interaktivno 2D in 3D računalniško grafiko. Z namenom upodabljanja na prenosljivih napravah, se iz OpenGL razvije OpenGL ES in WebGL (Web Graphich Library) [9]. WebGL predstavlja implementacijo OpenGL vmesnika do grafične strojne opreme z namenom uporabe v spletnih brskalnikih. WebGL temelji na specifikacijah OpenGL ES 2.0 in ohranja semantiko OpenGL ES, da poveča prenosljivost na mobilne naprave. OpenGL in WebGL vpeljeta neodvisnost od strojne opreme in platforme.

Pojavi se tudi ogrodje OpenCL [10], ki omogoča splošno namensko izvajanje na heterogenih platformah. Med heterogene platforme spadajo tudi grafične procesne enote (GPE), ki omogočajo visoko stopnjo paralelizma.

Razvoj je omogočil vedno učinkovitejšo uporabo tehnologije na veliko področjih. Uporabo računalniške grafike najdemo v filmih, igrah, simulacijah in oblikovanju, kjer z razvojem tehnologije vplivamo na stopnjo realizma in interaktivnosti. V znanosti jo srečamo na številnih področjih, kot so fizika,

kemija, medicina, biologija, geologija in drugih. Z vizualizacijo si pomagamo pri analizi, obdelavi in vizualizaciji velikih količin podatkov, na primer opazovanju polnih modelov, reševanju diferencialnih enačb dinamike tekočin, simulaciji porazdelitve temperature in simulaciji modeliranja vremena. Vizualizacijo srečamo tudi pri raziskovanju vesolja, razvoju materialov in razvoju zdravil.

1.2.1 Vizualizacija v fiziki delcev

Na področju fizike nam vizualizacija podatkov predstavlja pomembno orodje, predvsem na področju trkov osnovnih delcev visokih energij, s katerimi se ukvarja fizika osnovnih delcev. Fizika osnovnih delcev (angl. Paricle physics) je veja fizike, ki se ukvarja z osnovnimi gradniki snovi, sevanja in interakcijami med njimi. Pogosto se zanjo uporablja tudi izraz fizika visokih energij (angl. *high energy physics - HEP*).

Pri eksperimentih fizike delcev visokih energij vizualizacija predstavlja pomembno vlogo in je ključnega pomena. Pomaga nam kot orodje pri veliko aktivnostih in nalogah. Sodeluje pri razvoju in nadzorovanju detektorjev, pri ustvarjanju in rekonstrukciji dogodkov, pri simulaciji detektorjev in podatkovni analizi. Navsezadnje pa predstavlja močno orodje in pripomoček v izobrazbi in informativnosti. S primerno vizualizacijo si lahko pomagamo pri lažjem razumevanju problema, kvantnih pojavov in temeljito analiziramo sistem eksperimenta. Opazimo lahko pojave, ki bi iz surovih podatkov bili zakriti ali kontraintuitivni. Na ta način lahko natančneje opredelimo opazovani sistem eksperimenta.

V namene sodelovanja na področju računalništva in programske opreme v fiziki delcev visokih energij na mednarodni ravni je bila ustanovljena fundacija HSF (HEP Software Foundation). Fundacija opredeljuje številne dejavnosti, med katerimi je tudi vizualizacija in njej pripadajoča vizualizacijska skupina. Vizualizacijska skupina (angl. Visualization Working Group - VWG) združuje strokovnjake in razvijalce na področju vizualizacije v fiziki osnovnih delcev. Namen in cilj vizualizacijske skupine je raziskovanje,

načrtovanje in razvoj vizualizacije. Igra pomembno vlogo pri spodbujanju sodelovanja med eksperimenti fizike delcev visokih energij in raziskovalnimi skupinami kot pri razširjanju rezultatov v javnosti.

Obravnava vizualne predstavitve podatkov dogodkov z geometrijo detektorja z namenom raziskav, učenja in izobraževanja. Rešuje izzive, kot so izboljšave pregleda geometrije detektorja, simulacije in rekonstrukcije. Opredeljuje fizikalno analizo in učinkovito interaktivno upodabljanje na številnih platformah. Vizualizacijska skupina sodeluje tudi s preostalimi skupinami znotraj HSF, ki se zanašajo na vizualizacijo rezultatov.

Opredelitve in aspekti raziskovalnih področij so opisani v dokumentu, imenovanem *Community White Paper* (okrajšano CWP)[1]. CWP zajema veliko poglavij, med katerimi je tudi poglavje vizualizacije [2]. CWP predstavlja izzive pri analizi eksperimentov, ki jih srečamo v računalništvu in informacijski znanosti. Zajema poročila in opredelitve delovnih skupin štirinajstih področij.

Poglavje vizualizacije zajema in predstavlja vizualizacijo na področju eksperimentov HEP. Opisuje definicije, napredke in trenutno stanje vizualizacije podatkov. Predstavlja tri glavne tipe vizualizacije podatkov:

- interaktivne vizualizacije podatkov dogodka (prikazi dogodkov),
- vizualizacije statističnih podatkov za namene analize podatkov in
- prostorsko neodvisne vizualizacije podatkov.

Prikazi dogodkov (angl. event displays) omogočajo uporabniku raziskati podatke o dogodku. Ti so lahko vgrajeni v platforme eksperimentov ali pa prirejeni glede na prenosljivost in platformno neodvisnost. Prikazi dogodkov običajno vsebujejo podatke, ustvarjene ob eksperimentu in običajno tudi geometrijo, ki se nanaša na eksperiment. Pri tovrstnih vizualizacijah dogodkov opredelimo detektor in delce znotraj dogodka.

[1] `https://hepsoftwarefoundation.org/organization/cwp.html`

Vizualizacije statističnih podatkov omogočajo podatkovne analize. Tipično so to histogrami in grafi raznosa (angl. scatter plot), s katerimi opredelimo in karakteriziramo podatke. Tovrstne vizualizacije lahko vsebujejo podatke več dogodkov in niso tesno vezane na geometrijo detektorjev.

Prostorsko neodvisne vizualizacije podatkov zajemajo opise detektorjev, poddetektorjev in infrastrukture eksperimentov. Tovrstne vizualizacije se tipično pojavijo v obliki grafov in ponazoritev omrežij.

Prikazi dogodkov

Prikaze dogodkov lahko razdelimo na fazo dostopa do podatkov, fazo vizualizacije in fazo interaktivnosti.

Dostop podatkov je mogoč z dostopom do ogrodja, prirejenega eksperimentom HEP, ali preko vmesnih formatov podatkov, ki jih izvozimo iz ogrodja. Dostop podatkov preko ogrodja za eksperimente ponuja dostop do poljubnih podatkov v surovi obliki. Pomanjkljivost takega pristopa je platformna odvisnost, na kateri je zgrajeno samo ogrodje. Pri vizualizacijah si zato lahko pomagamo z integracijo poenostavljenih verzij ogrodja eksperimentov. Primer takega poenostavljenega ogrodja eksperimentov predstavlja CMSFireworks[2]. Dostop preko vmesnih formatov podatkov (npr. XML, JSON in CSV) ponuja razkorak med ogrodjem eksperimentov in lastno aplikacijo. S tem se ognemo patformni odvisnosti ogrodja eksperimenta in potencialno omogočimo manjšo kompleksnost in večjo prenosljivost razvitih orodij.

Prikaze dogodkov najpogosteje razvijamo za namizne aplikacije in prenosljive spletne aplikacije. Pojavijo pa se tudi nekatere mobilne aplikacije za telefone in aplikacije navidezne resničnosti (angl. virtual reality - VR).

Namizne aplikacije prikaza dogodkov največkrat temeljijo na C++ kodi in OpenGL aplikacijskem vmesniku, ki ponuja platformno neodvisnost in poln nadzor nad programsko in strojno opremo.

Nekatere aplikacije vizualizacije HEP uporabljajo OpenGL direktno z uporabo lastnih grafičnih pogonov. Primeri takih aplikacij so ATLAS Per-

[2] `https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookFireworks`

sint [6], ROOT EVE toolkit[3] in ALLICE Event Visualisation Environment[4]. V vizualizacijski skupini se je razvil sistem Geant4[1] z nameni vizualizacije geometrije, trajektorij in trkov. Glavni namen je opis detektorjev in simulacija trajektorij delcev in trkov. Sistem vsebuje komponente vizualizacije vmesnikov in vizualizacijskih pogonov za široko podporo grafičnih zmožnosti. Podpira različne grafične tehnologije, kot so sledenje žarku (angl. ray tracing) in OpenGL. Slabost takih pristopov je vzdrževanje in posodabljanje grafičnih pogonov.

Druge aplikacije uporabljajo višjenivojske knjižnice in grafične pogone, ki poenostavijo razvoj. Primer grafične knjižnice sta Open Inventor [20] in Coin3D[5], ki omogočata integrabilnost v C++ kodi in učinkovito delovanje. Aplikacija CMS Iguana[6] je bila ena izmed aplikacij, ki je uporabljala knjižnico Open Inventor. Aplikaciji, kot sta ATLAS VP1[7] [11] in LHCb Panoramix[8] pa uporabljajta knjižnico Coin3D. Slabost takih pristopov je odvisnost od zunanjih knjižic.

Spletne aplikacije temeljijo na WebGL implementaciji in imajo prednost enostavne prenosljivosti. Na voljo imajo tudi ogrodja kot je Three.js[9], ki poenostavi razvoj aplikacij. Primeri spletnih aplikacij so CMS iSpy[10] [15], ATLAS TRACER[11], ATLAS TADA [17] in LHCb Online[12].

Mobilne naprave so omejene s količino podatkov, ki jo lahko interaktivno obdelajo, ponujajo pa močno mobilnost in prenosljivost. Primer mobilne aplikacije predstavlja LHSee[13], ki je dogodke in podatke o dogodkih pred-

[3] https://root.cern.ch/eve
[4] http://alice-offline.web.cern.ch/Activities/Visualisation/index.html
[5] https://bitbucket.org/Coin3D/coin/wiki/Home
[6] http://iguana.cern.ch
[7] https://atlas-vp1.web.cern.ch/atlas-vp1
[8] http://lhcb-comp.web.cern.ch/lhcb-comp/frameworks/Visualization/
[9] https://threejs.org/
[10] http://cern.ch/ispy-webgl
[11] https://atlas-tracer.web.cern.ch/
[12] https://lhcb-public.web.cern.ch/lhcb-public/lbevent2/lbevent/
[13] https://www2.physics.ox.ac.uk/about-us/outreach/public/lhsee

vajala v živo. Mobilni aplikaciji sta tudi Camelia[14] in TEV[15]. Poleg tega spletna aplikacija CMS iSpy podpira možnost mobilnega načina uporabe.

Navidezna resničnost simulira prisotnost uporabnika v virtualnem okolju. Aplikacija CMS iSpy ima možnost takega načina uporabe, medtem ko aplikaciji ATLASrift[16] in Belle II VR[17] podpirata to možnost preko knjižnic navidezne resničnosti.

Prikazi dogodkov eksperimentov velikokrat zajemajo tudi informacije geometrije detektorjev. Geometrija je lahko direktno pridobljena iz ogrodja eksperimenta, ali pa se preko skript pretvori v druge oblike. Nekateri 3D modeli so narejeni v programski opremi modeliranja, kot je SketchUp [18] in izvoženi aplikacijam, ki so jih potrebovale.

Statistični podatki

Vizualizacija zajema tudi statistične vpoglede v eksperimente, ki se obravnavajo v obliki histogramov in grafov raznosa. V ta namen je CERN razvil analitično orodje in knjižnico ROOT [5], ki omogoča analizo podatkov. ROOT je postal najpogosteje uporabljeno orodje za izdelavo izrisov prikazov podatkov, grafike in celo prikazov dogodkov.

ROOT izrisovalno orodje omogoča možnost delovanja na namiznih napravah. Ima veliko funkcij, ki presegajo običajna standardna izrisovalna orodja. Poleg običajnih 2D in 3D izrisov, histogramov in grafov, omogoča izris 2D in 3D oblik. Podpira tudi gradnjo grafičnega uporabniškega vmesnika.

ROOT ima tudi JavaScript implementacijo za spletno analizo JSROOT[18].

Prostorsko neodvisni podatki

V HEP se pojavijo tudi podatki, ki so časovno in prostorsko neodvisni. Ti podatki so organizirani v drevesne strukture, ki jih najprimerneje vizualizi-

[14] `http://medialab.web.cern.ch/content/camelia`
[15] `https://gitlab.cern.ch/CERNMediaLab/TEV`
[16] `https://atlasrift.web.cern.ch/`
[17] `http://www1.phys.vt.edu/~piilonen/VR/`
[18] `https://root.cern.ch/js/`

ramo z drevesi, grafi in omrežnimi vizualizacijami. Primer tega je deskripcija detektorjev, ki opisuje vse sestavne dele detektorja eksperimenta v HEP. Sestavni deli so med seboj lahko prepleteni preko različnih relacij.

Pri tem imamo na voljo orodje, kot je GeoModel[4] in tehnične zasnove (npr. ATLAS inner detector: Technical Design Report, 1)[7].

GeoModel je knjižnica za opis komplekse geometrije detektorja. Glavna naloga je opis informacij geometrije detektorja za namene simulacije in rekonstrukcije. Omogoča dostop do surove geometrije detektorja in prav tako poljubnega podsistema surove geometrije. Podpira 3D vizualizacijo geometrije detektorja, detekcijo prekrivanja volumnov, rekonstrukcijo in simulacije, omogoča pa tudi podporo sledenja geometriji.

V tehničnih zasnovah je predstavljen opis detektorja. Vizualizacije detektorjev se tu pojavijo kot pripomoček k vpogledu opisa zgradbe in geometrije detektorja. Pojavijo se tudi oblikovanje načrtov, prikazi in simulacije dogodkov. Vizualizacija se pojavi tudi v obliki grafov in statistične lastnosti dogodkov.

1.2.2 Preostala dela

Področje diplome zajema podobnosti in aspekte publikacije z naslovom Fast segmentation, conversion and rendering of volumetric data using GPU [3]. V omenjenem članku je predstavljena dokazana konceptualna implementacija segmentacije hitrih volumetričnih podatkov, pretvorba v geometrijo poligonskega mrežnega modela in upodabljanje. Ta vsebina se navezuje na temo diplomske naloge, zlasti pri upodabljanju in obravnavi velike količine objektov z zelo visoko stopnjo vzporednosti na grafični procesni enoti. Sorodno delo je tudi magistrska naloga [16], ki opisuje način simulacije pretoka krvi skozi model krvnih žil. Predstavlja implementacijo odzivne interakcije in omogoča sistemsko neodvisnost in porazdeljeno procesiranje preko strežnika.

Podobnosti najdemo tudi v diplomi [14], ki obravnava postopke za vizualizacijo medicinskih volumetričnih podatkov v realnem času. Opisuje in primerja tri metode vizualizacije, ki temeljijo na metodi metanja žarkov.

Omenjene metode so tudi optimizirane za hitro izvajanje na grafičnih procesorjih. V delu je predstavljeno tudi nekaj inovativnih rešitev, ki pripomorejo k hitrejšem izrisovanju in omogočajo interaktivnost vizualizacije na počasnejših napravah.

Nekatere tematike se ujemajo tudi v diplomskem delu [12], ki opisuje spletno vizualizacijsko ogrodje Med3D, ki je bilo razvito v sklopu citirane diplomske naloge. Primarno se osredotoča na vizualizacije 3D mrežnih in volumetričnih medicinskih podatkov. Predstavlja implementacije različnih funkcionalnosti ogrodja, kot so upodabljanje, uporabniški vmesnik in oddaljeno sodelovanje.

1.3 Cilj

Vizualizacija eksperimentov v fiziki delcev lahko hitro postane problematična zaradi velikega števila elementov, še posebej kadar želimo zagotoviti visoko stopnjo interaktivnosti. Pri razvoju takšnega sistema vizualizacije nastopi naše sodelovanje, kjer naslavljamo in razrešujemo sledeče izzive:

- intuitivna vizualizacija prikaza podatkov osnovnih delcev in njihovih trajektorij v prostoru,
- realizacija platformno neodvisnega prikaza,
- visoka stopnja odzivnosti vizualizacije na interakcijo in
- možnost uporabe na vsakdanjih (tudi mobilnih) sistemih.

Cilj diplomske naloge je izdelava sistema, ki omogoča interaktivno vizualizacijo geometrije detektorja eksperimenta in vizualizacijo dogodkov interakcije delcev s posameznim detektorjem.

Ker posamezni dogodek v eksperimentu predstavlja veliko količino delcev, detekcij trkov delcev znotraj različnih delov geometrije detektorja eksperimenta in veliko rekonstruiranih trajektorij delcev, želimo realizirati učinkovito vizualizacijo. Hkrati želimo za posamezni delec, detekcijo trka ali pa

trajektorijo, ki pripada delcu, intuitivno vizualizirati izbrane parametre. Z uporabo upodabljanja v brskalniku (z uporabo WebGL-a) želimo realizirati platformno neodvisni prikaz in zagotoviti možnost uporabe na vsakdanjih sistemih. Za končni prikaz vizualizacije želimo upodabljanje v brskalniku združiti z upodobljevalnim sistemom na namenskem strežniku (razvitem z porabo knjižnice Vulkan).

Pri tem se želimo osredotočiti na intuitivno vizualizacijo prikaza osnovnih podatkov delcev in predvsem zagotoviti visoko stopnjo odzivnosti vizualizacije interakcije.

S tem delom želimo izboljšati trenutne prikaze dogodkov, ki se uporabljajo na področju fizike osnovnih delcev pri visokih energijah. Z uspešno realizacijo takega sistema bi omogočili tudi uporabo sistema kot orodje v pomoč tekmovalcem CERN-ovega izziva TrackML[19]. Izziv je bil razpisan v letu 2018 v CERN-u in objavljen na spletišču Kaggle. Cilj izziva je izdelava algoritma, ki hitro in učinkovito rekonstruira trajektorije pripadajočih delcev ob koliziji s pomočjo strojnega učenja. Razpisan je bil v namen raziskave učinkovitosti strojnega učenja v odkrivanju in karakterizaciji delcev. Potreba po tej raziskavi se je pojavila zaradi nadgradnje v CERN-u, ki bo povzročila eksperimente z zajemom še večje količine podatkov.

1.4 Struktura dela

Delo je razdeljeno v štiri poglavja. Uvodnemu poglavju 1 sledi poglavje 2, kjer predstavimo uporabljene tehnologije, zgradbo sistema in celoten postopek upodabljanja. Predstavili bomo pristope in metode, ki smo jih uporabili za reševanje problema optimizacije in intuitivnega prikaza vizualizacije. Temu sledi poglavje 3, ki zajema performančno analizo in rezultate analize. Tu sistem analiziramo in ovrednotimo. V zaključnem poglavju 4 povzamemo rezultate dela, podamo sklepne ugotovitve in opredelimo ideje in možna izhodišča za nadaljnji razvoj in delo.

[19] `https://www.kaggle.com/c/trackml-particle-identification`

Poglavje 2

Sistem za vizualizacijo trkov

Sistem za vizualizacijo želimo prirediti prikazom dogodkov. Cilj je opredeliti detektor in delce znotraj dogodka z uporabo primerne vizualizacije. S tem želimo omogočiti uporabniku, da lahko razišče podatke o dogodku.

Vizualizacija lahko implementira nadzor nad raznimi lastnostmi kot so barva, prosojnost in rezalne ravnine geometrije posameznega detektorja. Pri izrisovanju posameznih delcev in njihovih lastnosti pa dodamo še nadzor nad velikostjo oziroma debelino. Tu lahko velik izziv predstavlja implementacija nadzora nad debelino posamezne črte oziroma trajektorije. Če namreč želimo zagotoviti platformno neodvisnost prikaza, ne moremo uporabiti obstoječe funkcije WebGL-a, ampak, bomo morali realizirati drugo metodo, saj je obstoječa funkcija podprta zgolj na nekaj platformah.

Posamezno geometrijo detektorjev ter delce lahko iz scene odstranjujemo neodvisno od drugih. Vizualizacija prav tako omogoča nadzor nad posameznimi trajektorijami in detekcijami trkov delcev na detektorskih ploščah. Te lahko preko vmesnika poiščemo in izberemo, nakar se kamera postavi v položaj posamezne detekcije trka, delca ali trajektorije. Na tem mestu moramo zagotoviti intuitiven postopek, preko katerega bo kamera izpostavila določen trk delca ali trajektorijo. S tem lahko preučujemo naravo posameznih delcev, njihovo obnašanje, lastnosti in podobnosti s preostalimi delci in njihovimi trajektorijami.

Izziv predstavlja velika količina podatkov, ki jih želimo učinkovito vizualizirati z visoko stopnjo interakcije. Izziv predstavlja tudi sama oblika podatkov opisa detektorjev in lastnosti delcev, ki jih moramo obdelati in pretvoriti v ustrezno obliko. Prav tako oviro predstavlja realizacija obojestranske transparentnosti na vseh različnih detektorjih. Da zagotovimo pravilen transparenten izris, moramo biti pazljivi na vrstni red, kar je v primeru detektorja zaradi oblik in prekrivanja težka naloga.

S podobnim izzivom se bomo morali spopasti pri računanju podobnosti trajektorij, kjer je kompleksnost in težavnost problema velika.

Za uresničitev zadanih ciljev si lahko pri razvoju sistema za vizualizacijo pomagamo z obstoječimi spletnimi tehnologijami. Prav tako si lahko pomagamo z obstoječimi vizualizacijami dogodkov trkov delcev.

2.1 Uporabljene tehnologije

Ker je izmed ciljev realizacija vizualizacije na vsakdanjih sistemih, pride v poštev uporaba jezika JavaScript in spletnih tehnologij, preko katerih lahko vizualizacijo enostavno postavimo v vsakdanje računalniške sisteme. Z namenom optimizacije, da bi zadostili ciljem visoke stopnje odzivnosti vizualizacije na interakcijo, smo si podrobneje ogledali naslednja orodja in tehnologije, ki smo jih uporabili v okviru našega dela.

2.1.1 Knjižnica WebGL

Knjižnica WebGL [9] predstavlja aplikacijski programski vmesnik za programiranje grafičnega pospeševalnika v programskem jeziku JavaScript. Knjižnica nam omogoča 2D in 3D interaktivno upodabljanje v sodobnih brskalnikih. Program WebGL sestavlja kontrolna koda, napisana v kodi JavaScript, in senčilnik, ki je napisan v jeziku senčilnikov (angl. OpenGL Shading Language - GLSL). Jezik senčilnikov je podoben jeziku C ali C++ in se izvaja v računalniški grafični procesorski enoti (GPE).

Ker pa je knjižnica WebGl nizkonivojska in kompleksna, so se razvila razna orodja, kot je knjižnica Three.js.

2.1.2 Knjižnica Three.js

Knjižnica Three.js [8] predstavlja aplikacijski programski vmesnik, ki je namenjen upodabljanju animiranih 3D objektov v spletnih brskalnikih. Three.js uporablja knjižnico WebGL in omogoča izdelavo kompleksnih 3D objektov in animacij brez, da bi posegali po nizkonivojskih strukturah.

Knjižnica Three.js nam omogoča inicializacijo scene in definiranje objektov v sceni, kot so luči, kamere in geometrija.

2.1.3 Med3D

Med3D[1] [13] je odprtokodna spletna aplikacija, ki omogoča vizualizacijo 3D medicinskih podatkov. Omogoča prikaz 3D mrežnih in volumetričnih modelov neposredno v brskalniku. Aplikacija omogoča oddaljeno sodelovanje med uporabniki, ki si lahko izmenjujejo podatke, pogled, pripombe in lahko tudi komunicirajo z vgrajenim klepetom. Med3D platforma vsebuje hibridni način upodabljanja, ki omogoča odloženo upodabljanje (angl. deferred rendering). Združuje upodabljanje preko brskalnika z upodabljanjem na namenskem strežniku. Uporabniku omogoča interaktivno obdelavo in vizualizacijo podatkov. Aplikacija Med3D omogoča tudi razbremenitev uporabnikove naprave z izvajanje zahtevnejših izračunov na namenskem strežniku, kar zagotavlja visoko stopnjo interaktivnosti.

Med3D nam bo služilo kot ogrodje za končni prikaz vizualizacije, kjer bomo primerjali delovanje, testirali in analizirali performanco, napram sistemu zgrajenem preko okolja Three.js.

[1] `https://github.com/UL-FRI-LGM/Med3D`

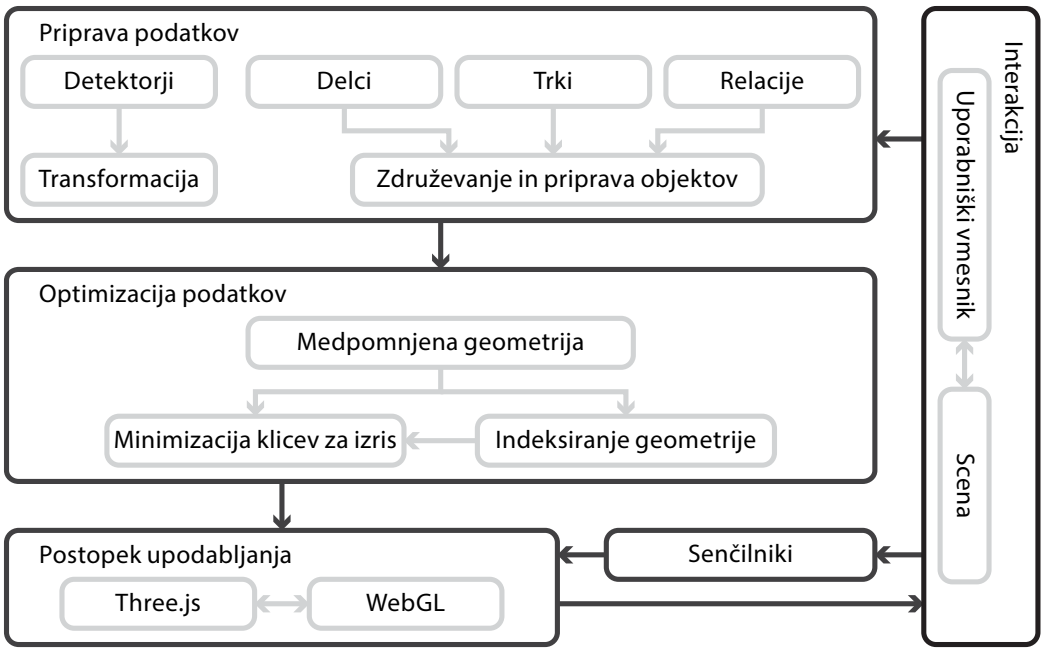
2.1.4 Knjižnica dat.GUI

Knjižnica dat.GUI[2] predstavlja preprosti grafični uporabniški vmesnik, ki je namenjen interaktivnem spreminjanju spremenljivk v jeziku JavaScript.

Za lažjo interakcijo sistema, smo z uporabo te knjižnice implementirali in prilagodili uporabniški vmesnik.

2.2 Ogrodje

Razviti sistem temelji na upodabljanju v spletnem brskalniku z uporabo tehnologije WebGL. Zaradi enostavnosti smo uporabili programsko ogrodje Three.js, namenjeno upodabljanju interaktivnih 3D upodobitev v spletnih brskalnikih v realnem času, ki abstrahira nizkonivojske funkcionalnosti WebGL-a. Shema delovanja sistema je prikazana na sliki 2.1, v nadaljevanju pa je predstavljen posamezni korak.



Slika 2.1: Na sliki je prikazana shema razvitega sistema.

[2] https://github.com/dataarts/dat.gui

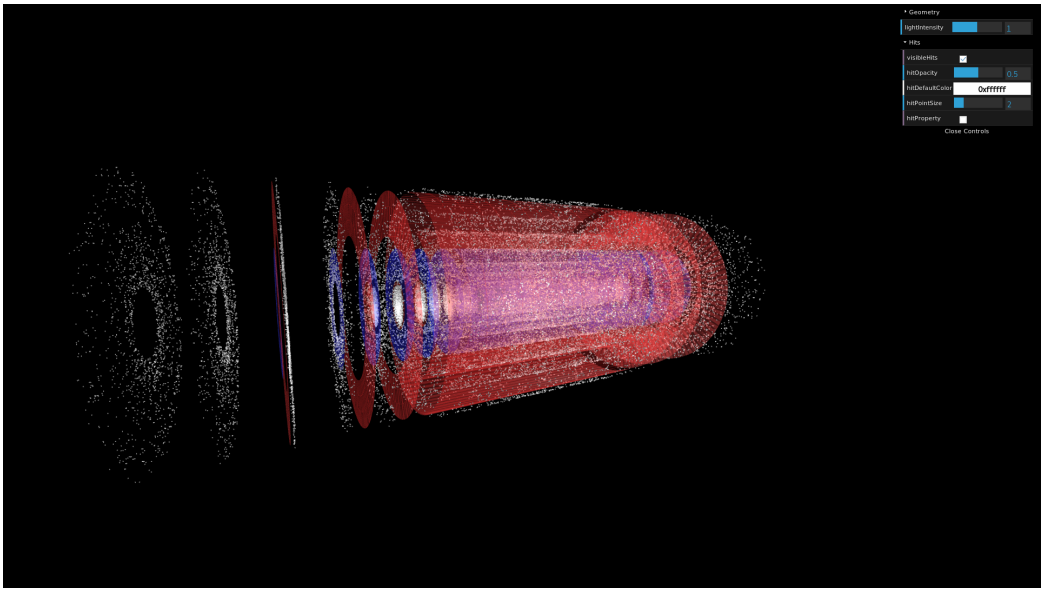
Sistem ob uporabnikovi zahtevi pripravi podatke fizikalnega eksperimenta trka delcev. Prebere informacije o detektorjih, iz njih naredi poligone in jih transformira v primerne koordinate prostora. Ob zahtevi za izbran dogodek sistem nato prebere informacije o delcih, njihovih trkih in medsebojnih relacijah. Sistem nato uredi podatke in jih glede na relacije združi v rekonstruirane trajektorije. V naslednjem koraku pripravi geometrijo in jo optimizira za učinkovito delovanje. Obenem pripravi material in senčilnike. Na tem mestu sistem pošlje podatke za upodabljanje v grafični cevovod. Rezultat upodobitve se prikaže v sceni, preko katere lahko s pomočjo uporabniškega vmesnika in nadzorom kamere interaktiramo z dogodkom eksperimenta.

Kot začetna osnova našega sistema smo si ogledali in povzeli primer vizualizacije spletne vizualizacije dogodka, ki temelji na vizualizaciji z uporabo spletnih tehnologij.

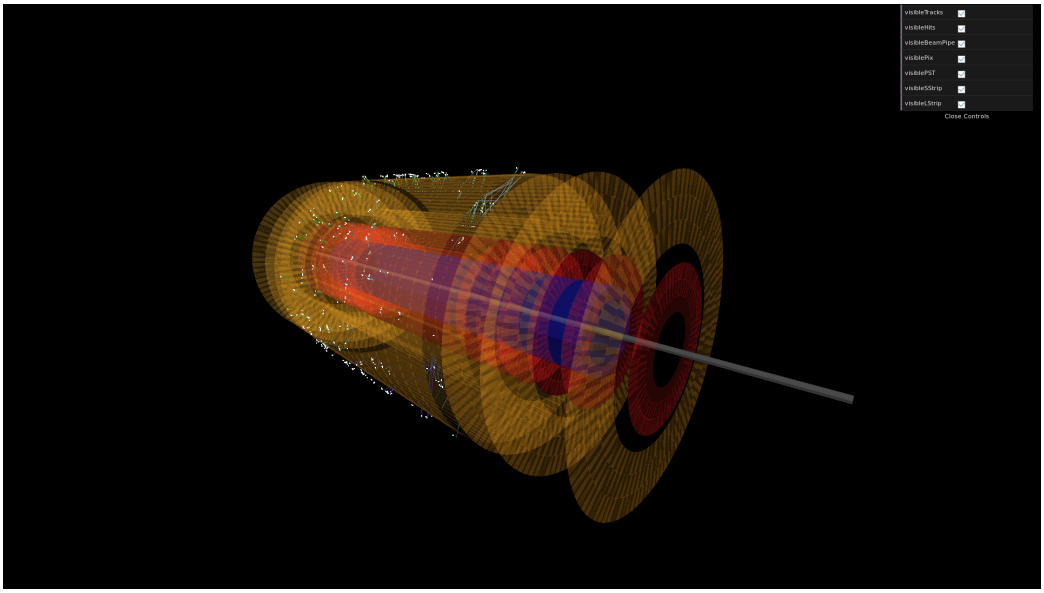
Omenjena osnova sistema nam omogoča prikaz posameznih delov detektorja, katerih opis preberemo iz datotek oblike OBJ. Omogoča pa tudi pregled primera pripravljenega dogodka, ki zajema detekcije trkov in trajektorije, katere preberemo iz prirejenih datotek tipa JSON. Pri tem nam nudi nadzor nad nekaterimi nastavitvami parametrov vizualizacije, kot so barva, prosojnost in velikost elementov v sceni.

Ta sistem ni optimiziran in počasi deluje na številnih platformah. Pri obdelavi običajnega dogodka z običajno količino podatkov na naši testni napravi sistem celo odpove, saj nam zmanjka glavnega in pomnilnika na disku (angl. swap). To povzroči, da se sistem kot proces ukine. Podrobnosti si lahko preberemo v poglavju 3.

Primer vizualizacije dogodka v izhodiščnem sistemu je prikazan na sliki 2.2, ki prikazuje detektorje in detekcije trkov. Slika 2.3 prikazuje primer vizualizacije izhodiščnega sistema, ki zajema detektorje, detekcije trkov in trajektorije.



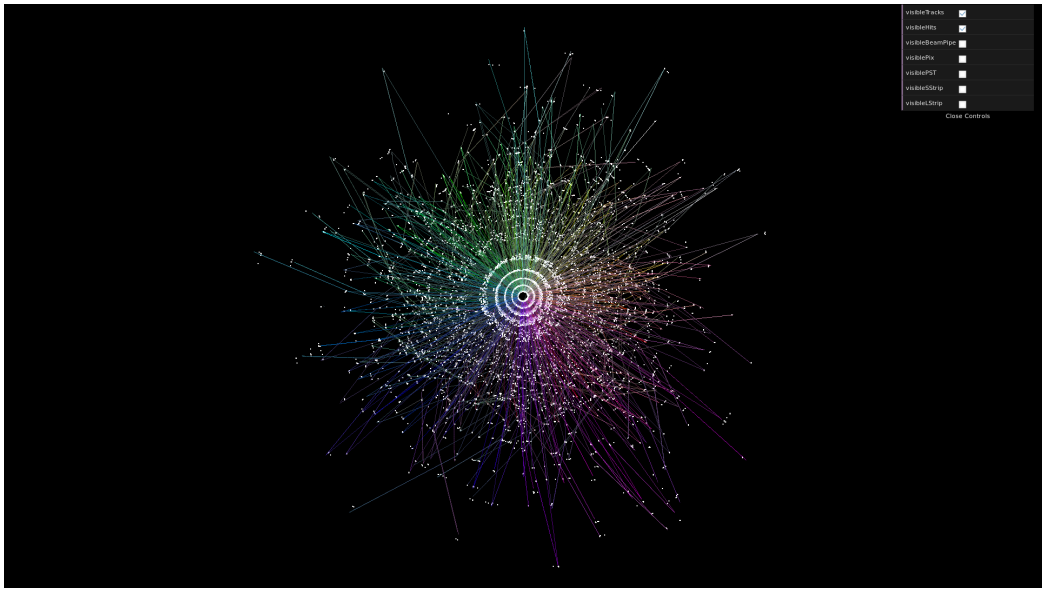
Slika 2.2: Primer vizualizacije detektorjev in detekcij trkov na izhodiščnem sistemu.



Slika 2.3: Primer vizualizacije detektorjev, detekcij trkov in trajektorij na izhodiščnem sistemu.

Zaradi narave prikaza transparentnih objektov v Three.js, oblike detektorjev in njihove medsebojne postavitve se pojavijo težave pri zagotavljanju pravilnega izrisa vseh elementov v sceni. Three.js obravnava prosojnost tako, da sprva upodobi vse neprosojne objekte z uporabo testa globine. Nato vključi zlivanje in upodobi prosojne objekte, urejene od zadnjega proti prvemu (glede na težišče).

Eno izmed takih anomalij lahko vidimo na sliki 2.3, ki se manifestira kot elementi dogodka (trki in trajektorije), ki niso vidni v notranjosti detektorjev. Poleg tega je prosojnost detektorjev vidna le v notranjost in ne tudi navzven. Posamezne trke in trajektorije lahko bolje vidimo na sliki 2.4, kjer smo zakrili geometrijo detektorjev v sceni in s tem omogočili upodobitev celotnih trajektorij.



Slika 2.4: Primer vizualizacije detekcij trkov in trajektorij na izhodiščnem sistemu.

2.3 Zajem, obdelava in priprava podatkov

Da sistem pravilno vizualizira fizikalni eksperiment trka delcev, potrebujemo natančne podatke, ki opredeljujejo fizikalni sistem. Potrebujemo podatke o geometriji detektorja in podatke o dogodku fizikalnega eksperimenta.

Podatke o geometriji detektorja in podatke o posameznih dogodkih v eksperimentih smo v začetku razvoja sistema vizualizacije pridobili s privatnega TrackML repozitorija. Nato smo te podatke pridobili s strani izziva TrackML, objavljenega na spletišču Kaggle. Podatki so na voljo v pomoč pri reševanju omenjenega izziva, namenjenega rekonstrukciji poti delcev razpisanega v letu 2018 v CERN-u. Zbirka podatkov vsebuje več samostojnih dogodkov, pri kateri vsak dogodek vsebuje simulirane meritve delcev, ki nastanejo pri trčenju protonskih snopov v Velikem hadronskem trkalniku v CERN-u.

Ker so se podatki in njihova struktura v začetku razvoja sistema še spreminjali, smo zajem in obdelavo podatkov posodabljali dokler se niso ustalili. Na tem mestu sledi opis postopka, ki ponazarja, kako pripravimo podatke za obdelavo na grafičnem procesorju.

2.3.1 Podatki o detektorju dogodka

Opisi detektorjev izhodiščnega sistema (v datotekah OBJ) niso bili povsem skladni s fizikalnimi eksperimenti. To lahko opazimo na sliki 2.2, kjer detektorji ne sovpadajo in se ne prilegajo z zaznanimi trki. Obenem lahko na sliki opazimo, da detektorji ne zajemajo celotnega eksperimenta. Zato smo se odločili geometrijo detektorjev naračunati iz surovih podatkov. Podatki o geometriji detektorjev se delijo na podatke o posameznih delih detektorjev, organiziranih v t. i. volumne, skupine in module, ki so običajno prilagojeni na zaznavanje določenega tipa delca in skupaj tvorijo celoten detektor v eksperimentu.

Podatki o geometriji detektorjev so na voljo kot atributi v datoteki tipa CSV, ki jih je potrebno pred uporabo pretvoriti v primerno obliko za izris (preden jih pošljemo v grafični cevovod).

Opis rekonstrukcije

Posamezni vnos v datoteki prestavlja centroid modula posameznega detektorja. Centroid moramo s primerno transformacijo, glede na tip detektorja, razširiti v pravokotni ali trapezoidni modul primernih dimenzij in postaviti na ustrezen položaj v prostoru.

To storimo tako, da za vsak centroid iz podatkov v datoteki zgradimo:

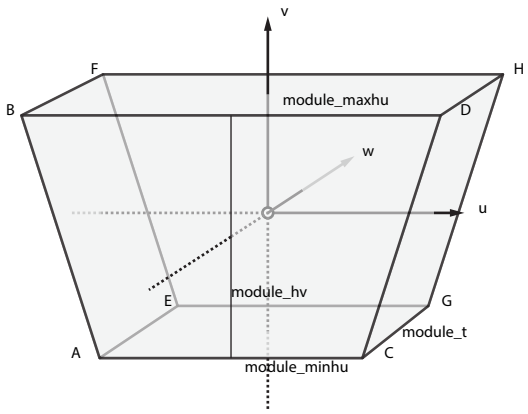
- rotacijsko matriko (***rotation_matrix***),
- vektor premika (***translation***) in
- položaj (***pos_uvw***).

Pri tem moramo položaj vsakega centroida (***pos_uvw***) še razširiti iz točkaste v pravokotno ali trapezoidno obliko modula (glede na tip detektorja). To storimo tako, da na mestu centroida (ki je v izhodišču lokalnega koordinatnega sistema) poračunamo odmike v prostoru, glede na podatke *module_minhu* (minimalna polovična dolžina), *module_maxhu* (maksimalna polovična dolžina), *module_hv* (polovična višina) in *module_t* (polovična širina).

Enačbe razširitve centroida v oglišča si lahko ogledamo na (2.1). Pri tem so A, B, ..., H poračunana vozlišča novonastalega detektorskega modula.

$$\begin{aligned}
A &= -module_minhu - module_hv + module_t \\
B &= -module_maxhu + module_hv + module_t \\
C &= +module_minhu - module_hv + module_t \\
D &= +module_maxhu + module_hv + module_t \\
E &= -module_minhu - module_hv - module_t \\
F &= -module_maxhu + module_hv - module_t \\
G &= +module_minhu - module_hv - module_t \\
H &= +module_maxhu + module_hv - module_t
\end{aligned} \tag{2.1}$$

Ponazoritev razširitve si lahko ogledamo na sliki 2.5.



Slika 2.5: Ponazoritev razširitve centroida v trapezoidni modul.

Rezultat je osem oglišč, katera lahko z uporabo transformacije pretvorimo iz lokalnega v globalni koordinatni sistem. To storimo z uporabo formule (2.2).

$$\boldsymbol{pos\_xyz} = \boldsymbol{rotation\_matrix} * \boldsymbol{pos\_uvw} + \boldsymbol{translation} \tag{2.2}$$

V enačbi (2.2) $\boldsymbol{rotation\_matrix}$ predstavlja rotacijsko matriko, ki jo zgradimo iz komponent rot_xu, rot_xv, rot_xw, rot_yu, rot_yv, rot_yw, rot_zu, rot_zv in rot_zw. Rotacijska matrika enačbe (2.2) je predstavljena na (2.3).

$$\boldsymbol{rotation\_matrix} = \begin{bmatrix} rot\_xu & rot\_yu & rot\_zu \\ rot\_xv & rot\_yv & rot\_zv \\ rot\_xw & rot\_yw & rot\_zw \end{bmatrix} \tag{2.3}$$

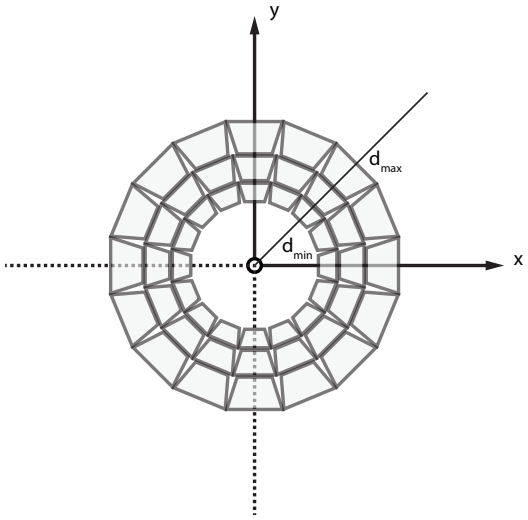
V enačbi (2.2) $\boldsymbol{pos\_uvw}$ predstavlja vektor (lokalnega) položaja oglišča, ki pripada pravokotnemu ali trapezoidnemu modulu. Oglišče zgradimo iz komponent u, v, w, ki smo jih naračunali v enačbah (2.1). Vektor premika je ponazorjen na (2.4).

$$\boldsymbol{pos\_uvw} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.4}$$

V enačbi (2.2) ***translation*** predstavlja vektor premika, ki ga zgradimo iz komponent cx, cy in cz. Vektor premika je ponazorjen na (2.5).

$$\boldsymbol{translation} = \begin{bmatrix} cx \\ cy \\ cz \end{bmatrix} \tag{2.5}$$

Za namene določanja barve v senčilnikih določimo tudi najbolj notranjo in najbolj zunanjo točko za vsak poddetektor. Zaradi cilindrične oblike poddetektorjev si s tem izračunom lahko pomagamo pri določanju presečišč poddetektorjev in trajektorij. Določitev notranje in zunanje točke je ponazorjena na sliki 2.6.



Slika 2.6: Prikaz določanja notranje in zunanje točke na poddetektorju.

Končne detektorske module s primerno orientacijo in indeksacijo vozlišč shranimo v medpomnjeno geometrijo in za potrebe osvetlitve poračunamo normale v ogliščih.

2.3.2 Podatki o delcih dogodka

Podatki o dogodku predstavljajo podatke o koliziji delcev, ki jih v tovrstnih eksperimentih opazujemo. Posamezen dogodek zajema sledeče podatke:

- podatke o delcih,
- podatke o trkih delcev,
- podatke o detektorskih celicah in
- podatke zlatega standarda (angl. ground turth), ki povezujejo trke s pripadajočimi delci.

Podatki o dogodkih eksperimenta znotraj detektorjev so prav tako zapisani v datotekah tipa CSV. Posamezni dogodek zajema informacije o delcih ob trku znotraj detektorja, informacije o vseh interakcijah delcev z detektorji in informacije, ki povezujejo interakcije s posameznimi delci. Informacije povežemo in shranimo v obliki, primerni za izris.

Priprava podatkov

Skupek datotek enega dogodka eksperimenta obsega približno 15 MB (megabajtov) prostora na disku. Z namenom minimizacije časa nalaganja datoteke asinhrono preberemo v glavni pomnilnik. Nato vsebino prebranih datotek za lažjo obdelavo pretvorimo v JavaScript objekt. Pri tem uporabljamo lastno funkcijo, ki smo jo priredili tovrstni nalogi.

Funkcijo smo priredili tako, da nam vse numerične vrednosti razčleni v obliko števila, podatke o identifikacijskih številkah pa kot niz znakov. To smo storili, ker se identifikacijska števila lahko pojavijo v zelo širokem obsegu, ki presega varno predstavitev celih števil v JavaScript.

JavaScript uporablja predstavitev števil v plavajoči vejici (angl. floating-point). Na ta način lahko varno predstavimo cela števila i v obsegu $-2^{35} < i < 2^{35}$. Za cela števila v obsegu $(-2^{35}, 2^{35})$ velja, da obstaja ena proti ena preslikava med matematičnim celim številom in celim številom, predstavljenim v jeziku JavaScript. V kolikor presežemo to območje, se lahko zgodi, da se več velikih celih števil zaradi zaokroževanja preslika v isto število.

Primer tega pojava sta celi števili $x = 229.684.405.646.397.440$ in $y = 229.684.405.646.397.441$, ki se v JavaScript predstavitvi obe preslikata v enako celo število $z = x = 229.684.405.646.397.440$.

Pojav je obravnavan v definicijah ECMAScript (okrajšano *ES*)[3], ki v 6. izdaji izpostavlja sledeče omejitve celih števil:

```
Number.MAX_SAFE_INTEGER = Math.pow(2, 53)-1;
Number.MIN_SAFE_INTEGER = -Number.MAX_SAFE_INTEGER;
```

Na tem mestu pregledamo celotne opise vseh zajetih datotek, ki predstavljajo dogodek trka delcev. Vse omenjene informacije med seboj strukturirano povežemo in pripravimo v ustrezno obliko predstavitev delcev, trkov in njihovih trajektorij, ki je primerna za upodabljevalni cevovod.

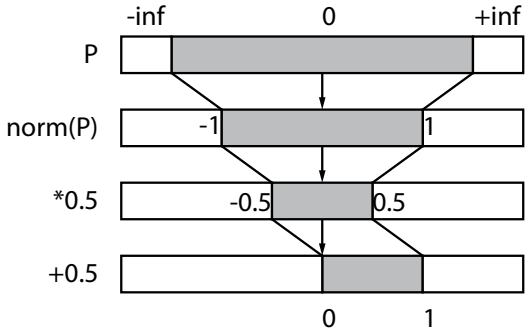
Iz objektov za vsak element eksperimenta razberemo različne podatke, kot so identifikacijsko število, položaj in podatek o gibalni količini. Položaj predstavlja prostorski odnos v globalnem koordinatnem sistemu. V prostoru $\mathbb{R}^3$ ga določimo z vektorjem treh komponent x, y in z. Gibalna količina $\boldsymbol{P}$ je fizikalna količina, ki je enaka zmnožku mase m in hitrosti $\boldsymbol{v}$ točkastega telesa ($\boldsymbol{P} = m * \boldsymbol{v}$). Predstavimo jo z vektorjem v prostoru. Posamezne komponente vektorja gibalne količine name povedo, koliko gibalne količine ima telo v posamezni smeri prostora x, y in z.

Ker želimo gibalno količino uporabiti za določanje barve v senčilnikih, jo moramo predstaviti v barvnem prostoru RGB. Prostor lahko ponazorimo s kocko v prostoru $\mathbb{R}^3$. Predstavlja nabor barv s kombinacijami komponent rdeče, zelene in modre. Posamezna komponenta RGB prostora se nahaja na intervalu $[0, 1]$. Ker gibalna količina lahko presega ta interval, jo preslikamo v primeren obseg.

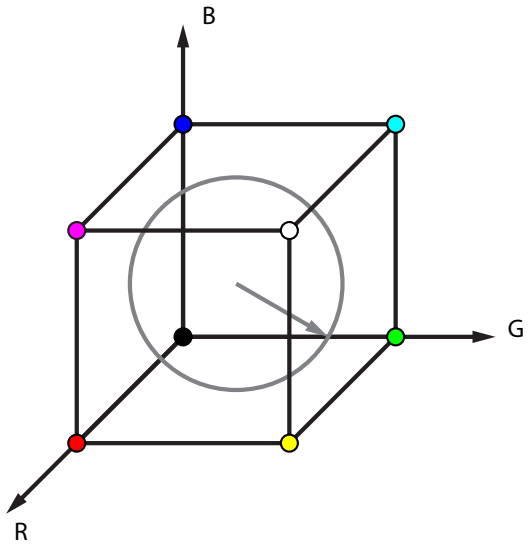
Da dosežemo primerno preslikavo gibalne količine v vrednosti prostora RGB, jo najprej normaliziramo. S tem omejimo, da posamezne komponente gibalne količine ne bodo presegale intervala $[-1, 1]$. S tem bodo vektorji gibalne količine obsegali prostor na površini sfere, ki je od izhodišča oddaljena za natanko 1. Gibalno količino nato pomnožimo z 0.5, kar prepolovi vrednosti na interval $[-0.5, 0.5]$. Vrednost gibalne količine še premaknemo za 0.5 na interval $[0, 1]$, ki je primeren ponazoritvi z barvo v prostoru RGB.

[3] `https://en.wikipedia.org/wiki/ECMAScript#cite_ref-1`

Preslikava gibalne količine $\boldsymbol{P}$ je ponazorjena na sliki 2.7. Rezultat je območje gibalne količine, ki sovpada s prostorom RGB (prikazano na sliki 2.8).



Slika 2.7: Ponazoritev povzetka celotne spremembe obsega vektorjev gibalne količine na primeren interval za predstavitev v RGB prostoru.



Slika 2.8: Ponazoritev vektorjev gibalne količine, ki obsegajo intervale RGB prostora.

Splošno formulo preslikave si lahko ogledamo na (2.6) in formulo preslikave z intervala $[-1, 1]$ na interval $[0, 1]$ na (2.7). Pri tem a_{low} in a_{high}

predstavljata območje intervala katerega želimo preslikati, a_{min} in a_{max} pa predstavljata območje ciljnega intervala.

$$f(a) = a_{min} + (a - a_{low}) * \frac{a_{max} - a_{min}}{a_{high} - a_{low}} \tag{2.6}$$

$$f(a) = (a + 1) * \frac{1}{2} \tag{2.7}$$

$$f(a) = a * 0.5 + 0.5$$

S tem tudi razbremenimo grafično procesno enoto, ker izračun opravimo le enkrat na strani centralne procesorske enote. Identifikacijsko število enolično določa posamezni delec. Ker to število lahko presega obseg 32 bitne predstavitve atributov (ki je optimizirana za delovanje na GPE), naredimo preslikavo. Preslikava omogoča, da je največje zastopano število enako številu vseh preslikanih elementov (kar je primerno za predstavitev z 32 biti). Enostavna preslikovala funkcija je ponazorjena v sledečem segmentu:

```
let longIDs = {longID1, longID2, ..., longIDn};
let shortIDs = {};
for(let i = 0; i < longIDs.length; i++){
      shortIDs[longIDs[i]] = i;
}
```

Informacije o delcih ob branju datoteke shranimo v polje. Shema 2.9 prikazuje združevanje objektov delcev v polje.

Slika 2.9: Zgradba polja objektov delcev.

Za vsak delec razberemo podatke o njegovem položaju, gibalni količini in njegovem identifikacijskem številu. Položaj delca določimo ga z vektorjem treh komponent vx, vy in vz. Gibalno količino predstavimo z vektorjem treh komponent px, py in pz.

Atribute delcev na tem mestu shranimo v medpomnjeno geometrijo.

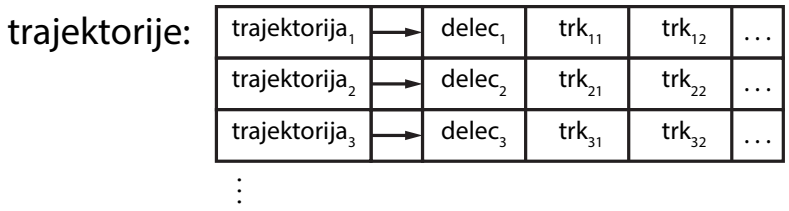
Informacije o trkih branju datoteke shranimo v polje. Shema 2.10 prikazuje združevanje objektov trkov v polje.

Slika 2.10: Zgradba polja objektov trkov.

Za predstavitev trkov razberemo podatke o njihovih položajih, njihovi identifikacijskih številih in identifikacijskih številih delcev, ki pripadajo trku (povzročitelj trka). Položaj trka določimo z vektorjem treh komponent x, y in z. Identifikacijsko število trka in identifikacijsko število delca preslikamo v primeren obseg na enak način kot prej.

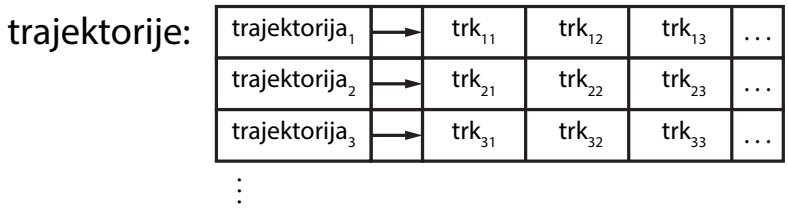
Atribute trkov na tem mestu shranimo v medpomnjeno geometrijo.

Trajektorije sestavimo tako, da pregledamo celoten opis datoteke zlatega standarda, ki opisuje preslikavo med delci in njihovimi trki. Za vsak vnos v datoteki zlatega standarda preverimo, kateremu delcu pripada trk. Informacije o trku nato shranimo v polje pripadajočega delca. Na ta način zberemo vse informacije trkov za posamezni delec na enem lokaliziranem mestu. Shema 2.11 prikazuje združevanje trkov v trajektorije, ki pripadajo delcem.



Slika 2.11: Zgradba polja objektov trajektorij s trki.

Vsaki trajektoriji na začetek dodamo še pripadajoči začetni delec, kar ponazarja slika 2.12.



Slika 2.12: Zgradba polja objektov trajektorij z delci in trki

Za predstavitev zgrajenih trajektorij razberemo podatke o položaju posameznih vozlišč, gibalni količini delca v vozliščih in identifikacijsko število trajektorije (ki je enako identifikacijskem številu delca). Položaj posameznega oglišča trajektorije določimo z vektorjem komponent tx, ty in tz. Gibalno količino v ogliščih trajektorije opišemo z vektorjem tpx, tpy in tpz.

Atribute na tem mestu indeksiramo in shranimo v medpomnjeno geometrijo.

2.4 Optimizacija izrisa

Z optimizacijo izrisa zagotovimo potrebi po visoki stopnji interaktivnosti vizualizacije in s tem možnosti uporabe na vsakdanjih sistemih. Pri optimizaciji smo skušali zmanjšati prenose na grafični pospeševalnik in omogočiti polno izkoriščenost. Uporabili smo tehnike medpomnjenja geometrije in indeksiranja geometrije, obenem pa smo skušali minimizirati število klicev za izris.

2.4.1 Medpomnjena geometrija

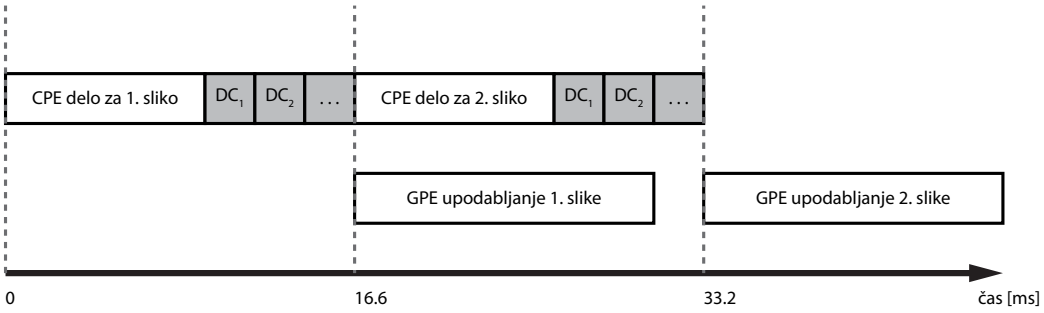
Z namenom učinkovite predstavitve geometrije (poligonov, črt ali točk) smo uporabili medpomnjeno geometrijo, ki je nekoliko težja za uporabo in zahteva nižjenivojsko razumevanje. Medpomnjena geometrija je v splošnem hitrejša in omogoča hitrejši izris. Vsi atributi so shranjeni v medpomnilnikih (namesto direktno kot objekti) kar skrajša čas prenosa podatkov na grafično kartico.

Omenjeni pristop smo lahko uporabili pri predstaviti detektorjev in predstavitvi delcev trkov in trajektorij. Atribute, kot so podatki o položajih, normalah, barvah, in preostale atribute smo shranili v medpomnilnike. Ker smo uspeli podatke o geometriji v celoti na ta način predstaviti, smo s tem skrajšali čas prenosa na GPE in porabo pomnilnika.

2.4.2 Minimizacija klicev za izris

Ker v opisu posameznega dogodka trka nastopa velika količina informacij (npr. 12.263 delcev, 120.939 trkov in 10.566 trajektorij), smo se pri optimizaciji najbolj osredotočili na minimizacijo klicev za izris (angl. draw calls), kar se je izkazalo za pomemben dejavnik in je razvidno iz rezultatov v poglavju 3.

Klic za izris predstavlja strnjeno informacijo o vozliščih, senčilnikih, medpomnilnikih, objektih, teksturah, ipd. znotraj t. i. vektorjih stanj (angl. state vectors). CPE pripravi in prevede vektorje stanj, ki jih posreduje na GPE. Če moramo pripraviti veliko klicev za izris, to lahko privede, da porabimo preveč časa za pripravo. V tem primeru mora GPE čakati na CPE, da pripravi vse podatke, kar privede do nedejavnega delovanja in zmanjšane učinkovitosti. GPE na ta način ostane neizkoriščena. Ponazoritev klicev za izris si lahko ogledamo na sliki 2.13.



Slika 2.13: Na sliki je ponazorjeno delo CPE, ki pripravi nekaj klicev za izris (DC).

Detektorji

Posamezna geometrija modula detektorja je predstavljena z zaporedjem osmih vozlišč, ki skupaj tvorijo trapezoidni ali pravokotni modul.

Geometrijo posameznih detektorjev upodobimo z uporabo trikotniških primitivov (WebGL primitiv gl.TRIANGLES), ki za skupke treh vozlišč celotne geometrije priredi trikotnike. Ker smo uspeli celotno geometrijo detektorja shranili v skupni medpomnilnik, nam to omogoča, da celotno geometrijo posameznega detektorja upodobimo z enim klicem za izris.

Delci in trki

Geometrija posameznega delca in trka je predstavljena z vozliščem.

Podatke vseh delcev v začetnih položajih in položaje detekcij trkov upodobimo kot točke (WebGL primitiv gl.POINTS), ki za vsako specificirano vozlišče celotne geometrije izriše matematično točko. Vse delce in vse trke lahko shranimo v enoten medpomnilnik delcev in enoten medpomnilnik trkov. Na ta način upodobimo celotno množico začetnih položajev delcev in vse detekcije trkov vsako z enim klicem za izris.

Trajektorije

Geometrija posamezne trajektorije je predstavljena z zaporedjem N vozlišč. Vozlišče $i = 1$ predstavlja prvo vozlišče in $i = N$ zadnje vozlišče.

V primeru, da vsako trajektorijo shranimo v svoj medpomnilnik in jih skušamo izrisati, se bo vsaka trajektorija izrisala v svojem klicu za izris. Vsaka trajektorija namreč zahteva drugačen nabor stanja atributov (položaji, gibalna količina ...), kar povzroči dodatne klice za izris. Če želimo vse trajektorije upodobiti z enim klicem za izris, jih moramo skupaj shraniti v mepodmnilnik, nad katerim izvedemo izris.

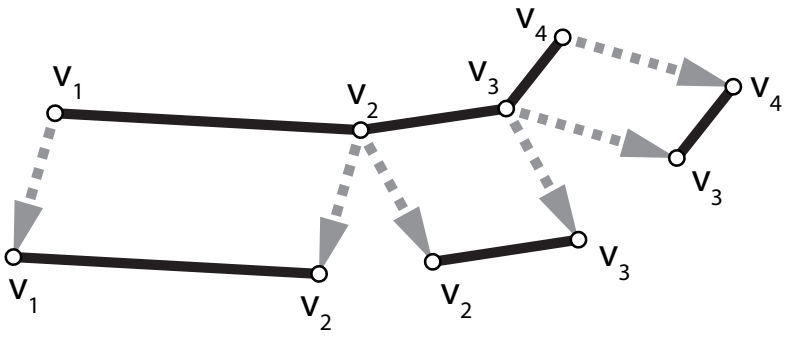
Upodobitev trajektorij kot trak linij (WebGL primitiv gl.LINE_STRIP) povzroči, da se različne trajektorije v medpomnilniku med seboj povezujejo. Problem se pojavi, ker se poleg samih trajektorij izriše tudi povezava med

dvema ločenima trajektorijama (zadnje vozliče prve trajektorije $i_1 = N$ se poveže s prvim vozliščem druge $i_2 = 1$).

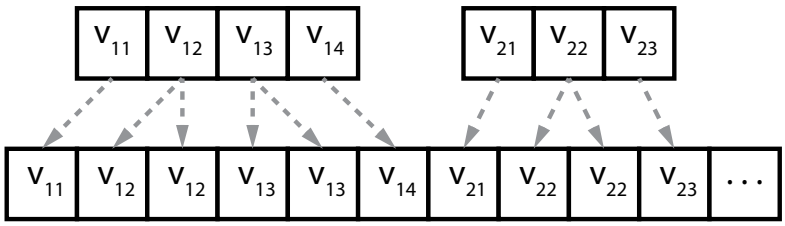
Na tem mestu bi si lahko pomagali z Three.js metodo `drawRange` ali metodo `drawGroups`, vendar bi povzročili dodatne klice za izris za vsako trajektorijo. Metoda `drawRange` določi, kateri del geometrije naj se upodobi. Metoda `drawGroups` razdeli geometrijo v skupine, ki se izrišejo vsaka v svojem klicu za izris.

Če se držimo metodologije paralelizma in skušamo vse trajektorije razdeliti na elementarne dele, pridemo do rešitve, ki združuje omogočanje učinkovitega indeksirana in omogoča upodobitev z enim klicem za izris.

Posamezno trajektorijo razdelimo na segmente daljic (slika 2.14). Segmente daljic vseh trajektorij shranimo kot skupno geometrijo v medpomnilniku na način, ki nam omogoča ločevanje med trajektorijami, obenem pa povezuje daljice iste trajektorije (slika 2.15).



Slika 2.14: Na sliki je prikazana zgradba posamezne trajektorije iz posameznih daljic.

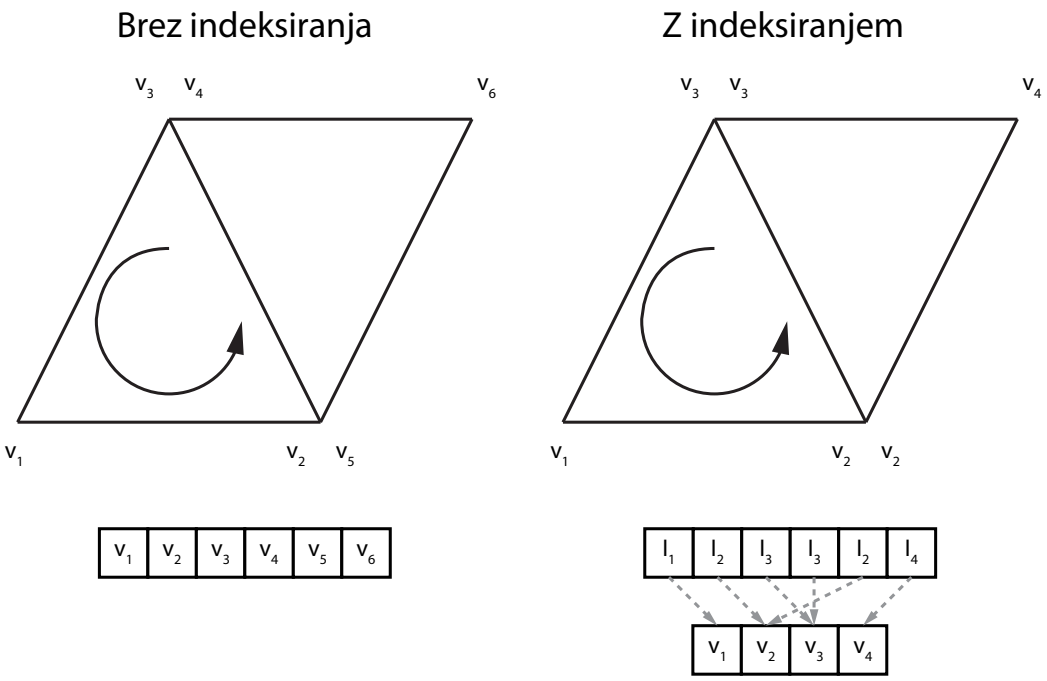


Slika 2.15: Ponazoritev načina shranjevanja trajektorij v medpomnilnik, ki omogoča ločevanje med trajektorijami.

Množico vseh trajektorij upodobimo kot verigo črt (WebGL primitiv gl.LINES), ki za vsak par dveh točk izriše linijo. Na ta način lahko vse trajektorije upodobimo zgolj z enim klicem za izris.

2.4.3 Indeksiranje geometrije

Za dodatno pohitritev smo za predstavitev geometrije (detektorjev in trajektorij) uporabili indeksirano obliko zapisa. Indeksirana oblika zapisa geometrije omogoča uporabo že obstoječih atributov vozlišč. Medpomnilnik indeksov vsebuje cela števila, ki naslavljajo atribute v medpomnilniku (položaj, barva, UV koordinate, normale ...). V splošnem to pomeni učinkovitejše delovanje, manj dela za senčilnike in manjšo porabo pomnilnika. Ponazoritev običajne in indeksirane geometrije si lahko ogledamo na sliki 2.16.



Slika 2.16: Na sliki je prikazana primerjava običajne in indeksirane geometrije.

S tem zmanjšamo porabo pomnilnika, kot tudi skrajšamo čas prenosa na GPE. Manjša poraba pomnilnika na GPE velikokrat pomeni učinkovitejše delovanje. S tem zmanjšamo prenose na grafičnem pospeševalniku, kjer pasovna širina pomnilnika predstavlja eno izmed ozkih grl v učinkovitosti izvajanja.

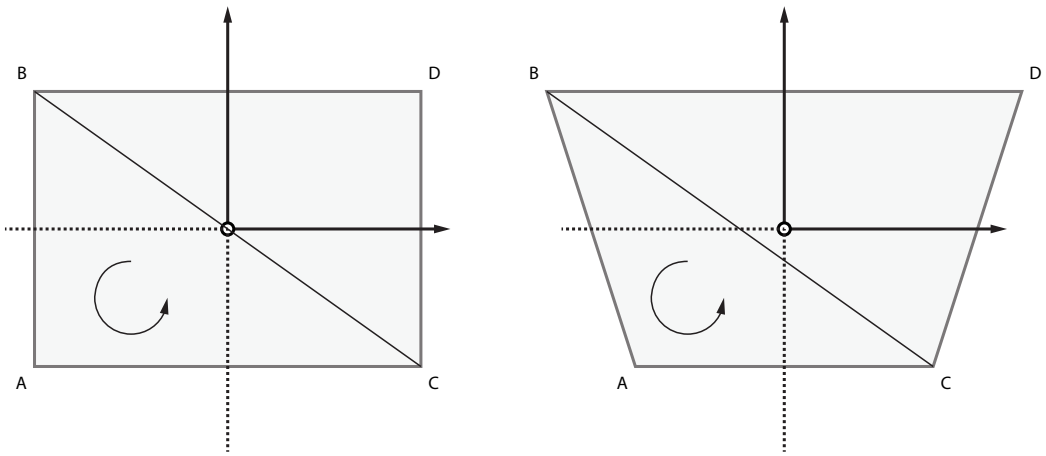
Z uporabo indeksiranja na GPE omogočimo tudi možnost uporabe predpomnjenja (Post Transform Cache[4]). Post Transform Cache je del strojne opreme na upodabljevalnem cevovodu grafične kartice. V primeru uporabe indeksiranja ta skuša ugotoviti uporabo enakih vozlišč in njihovih atributov. Če uspe najti že procesirano vozlišče, preskoči branje atributov vozlišča in procesiranje vozlišča (senčilnik vozlišč). Na tem mestu uporabi kopijo že transformiranih (angl. post-transform) podatkov na izhodu (angl. output stream), kar prihrani procesiranje vozlišč. V najboljšem primeru se vsako vozlišče procesira le enkrat. S tem omogočimo optimizacijo nad prekrivajočimi

[4] `https://www.khronos.org/opengl/wiki/Post_Transform_Cache`

podatki na GPE.

Detektorji

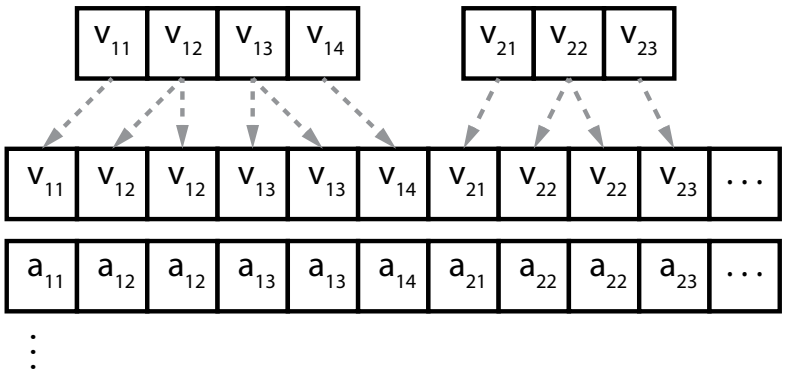
Indeksiranje uporabimo nad geometrijo detektorjev, kjer so posamezni moduli detektorjev zgrajeni iz pravokotnih ali trapezoidnih teles. Tu si vsako vozlišče modula deli več trikotnikov. Primer indeksiranja detektorjev je na sliki 2.17.



Slika 2.17: Slika ponazarja uporabo indeksiranja pri pravokotnih in trapezoidnih modulih detektorjev.

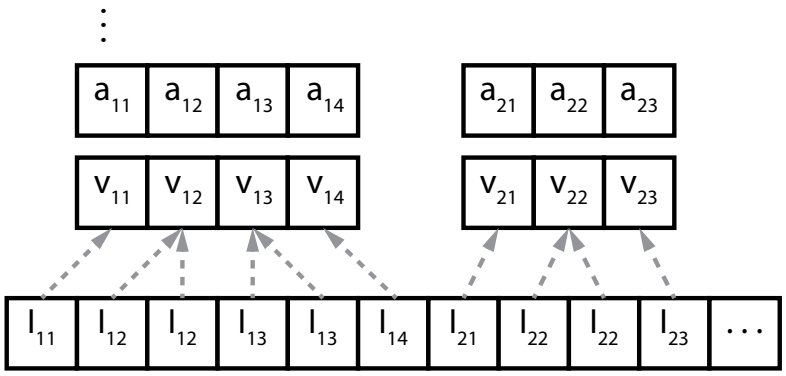
Trajektorije

Indeksiranje uporabimo nad geometrijo trajektorij, kjer se podvajajo vozlišča med segmenti daljic. Vsaka trajektorija ima N vozlišči. Vsebuje začetno in končno vozlišče, ki imata namen ločevanja med preostalimi trajektorijami znotraj skupnega medpomnilnika in $N-2$ vozlišč, ki se podvojijo in jih lahko indeksiramo. Geometrija brez indeksiranja je ponazorjena na sliki 2.18.



Slika 2.18: Slika prikazuje neindeksirano geometrijo trajektorij v medpomnilnikih.

Ker se vozlišča znotraj trajektorije podvajajo, izvedemo indeksiranje geometrije, pri čemer obdržimo relacije daljic v posameznih trajektorijah. Primer indeksiranja trajektorij vidimo na sliki 2.19. S tem pridobimo še dodatno pohitritev. Rezultate doprinosa indeksiranja si lahko ogledamo v poglavju 3.



Slika 2.19: Slika prikazuje indeksiranje geometrije trajektorij.

2.5 Osvetljevanje in senčenje

Pojem osvetljevanje (angl. lighting) v računalniški grafiki pomeni izračun lastnosti osvetlitve predmeta.

Poznamo globalno in lokalni način osvetljevanja. Globalne tehnike podajajo večjo stopnjo realizma, vendar so računsko bolj zahtevne. Lokalne tehnike so poenostavljene in trenutno bolj primerne za interaktivno upodabljanje.

Model lokalnega osvetljevanja lahko opredelimo s komponentami:

- ambientna svetloba (angl. ambient) k_a,
- emisijska svetloba (angl. emission) k_e,
- razpršeni odboj (angl. diffuse) k_d in
- zrcalni svetloba (angl. specular) k_s.

Za izračun osvetlitve komponente združimo in za vsako luč (i) v sceni po formuli (2.8) poračunamo prispevke za barvo c.

$$\boldsymbol{c} = \boldsymbol{c_a k_a} + \boldsymbol{k_e} + \sum_i \boldsymbol{c_i}(\boldsymbol{k_d}(\boldsymbol{l_i} \cdot \boldsymbol{n}) + \boldsymbol{k_s}(\boldsymbol{h_i} \cdot \boldsymbol{n})^s) \tag{2.8}$$

Pojem senčenje (angl. shading) pomeni tehniko obarvanja in izrisa osvetljenega predmeta.

Tipične tehnike senčenja so plosko (angl. flat), Gouradovo in Phongovo. Pri ploskem senčenju računamo osvetlitev celotnega poligona, z izračunom v enem oglišču ali pa povprečjem več oglišč poligona. Ta način senčenja je hiter, a kot rezultat poda nezvezen izgled. Pri Gouradovem senčenju računamo osvetlitev v vseh ogliščih poligona. Notranjost poligona interpoliramo med barvami v ogliščih. Pri Phongovem senčenju računamo osvetlitev v posameznih točkah — fragmentih. To dosežemo tako, da interpoliramo normale in nato poračunamo osvetlitev glede na izbrani model osvetljevanja. Phongovo senčenje je nekoliko počasnejše, a poda mehkejši izgled.

2.5.1 Detektorji

Osvetljevanje geometrije detektorja izračunamo z uporabo Lambertovega (difuznega) modela osvetljevanja[5], ki opredeljuje razpršeni odboj (angl. diffuse).

[5] `https://en.wikipedia.org/wiki/Lambertian_reflectance`

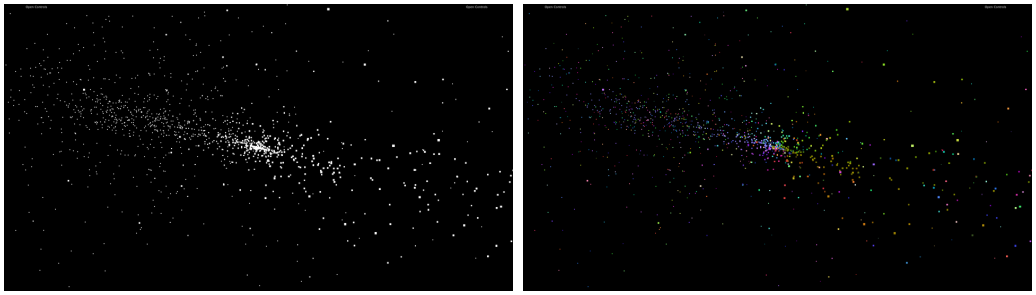
Lokalni model osvetljevanja s tem poenostavimo v formulo (2.9).

$$\boldsymbol{c} = \boldsymbol{k_e} + \sum_i \boldsymbol{c_i}(\boldsymbol{k_d}(\boldsymbol{l_i} \cdot \boldsymbol{n})) \tag{2.9}$$

Senčenje izvedemo z Gouraudovo tehniko senčenja[6], kar pomeni, da osvetlitev računamo na ogliščih in nato barvo interpoliramo v notranjost poligonov. S tem dosežemo primeren izgled za našo vizualizacijo in obenem učinkovito izvajanje.

2.5.2 Delci

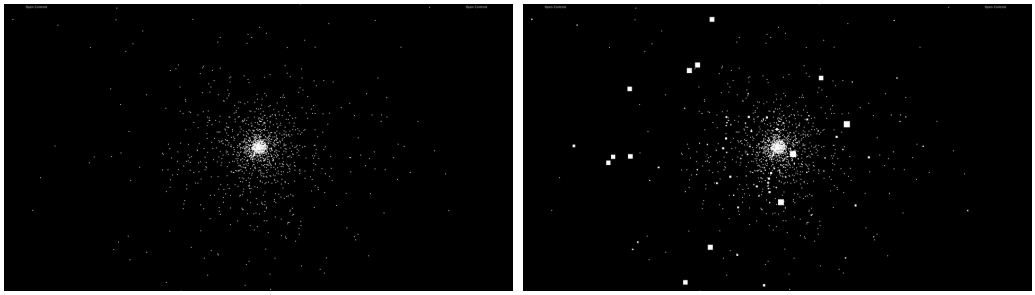
Senčenje delcev smo izvedli z nadzorom nad velikostjo, barvo in prosojnostjo. Pri tem lahko barvo določimo glede na atribut gibalne količine delca ali pa priredimo enotno barvo vsem delcem. Primer izračuna barve si lahko ogledamo na sliki 2.20.



Slika 2.20: Na levi strani so delci, kjer jim določimo enotno barvo. Na desni strani so delci, kjer jim barvo določimo glede na atribut gibalne količine.

Ker je velikost upodobljenih primitivov delcev ne glede na oddaljenost od kamere vedno enaka, smo senčilnik opremili z dinamičnim izračunom velikosti. Velikost upodobljenih točk, s katerimi ponazorimo delce, izračunamo glede na oddaljenost od kamere. Tako delce, ki so bližje kameri, izrišemo večje kot bolj oddaljene delce. Primer dinamične spremembe velikosti delcev glede na oddaljenost od kamere si lahko ogledamo na sliki 2.21.

[6] `https://en.wikipedia.org/wiki/Gouraud_shading`

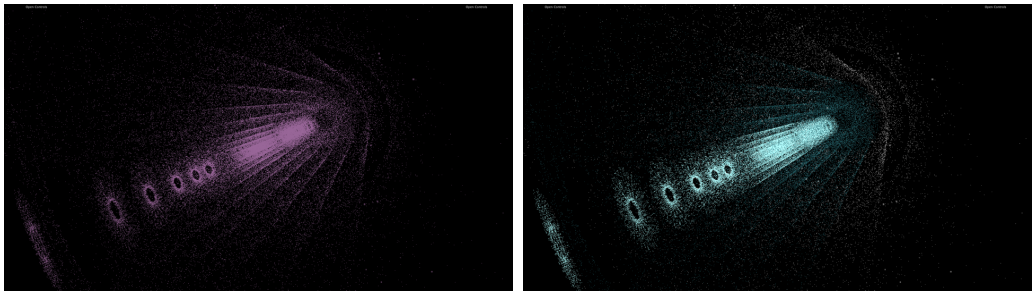


Slika 2.21: Slika prikazuje delce v začetnih položajih. Na levi strani so delci, kjer njihove velikosti ne spreminjamo ne glede na oddaljenost od kamere. Na desni strani so delci, kjer njihove velikosti spreminjamo glede na oddaljenost od kamere.

V primeru, da smo v sceni izbrali določen element dogodka za opazovanje, to dodatno poudarimo s spremembo velikosti točke delca, barve in prosojnosti.

2.5.3 Trki

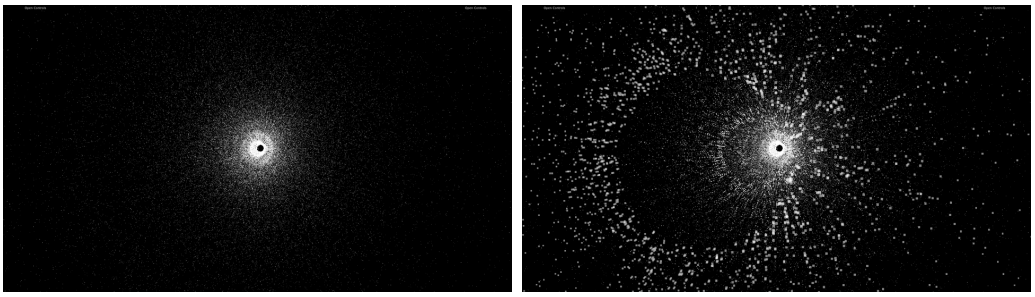
Tudi senčenje zaznanih trkov delcev smo izvedli z nadzorom nad velikostjo, barvo in prosojnostjo. Barvo določimo glede na tip detektorja, kjer je bil delec zaznan, ali pa priredimo enotno barvo. Primer izračuna barve si lahko ogledamo na sliki 2.22.



Slika 2.22: Na levi strani so trki, kjer jim določimo enotno barvo. Na desni strani so trki, kjer jim barvo določimo glede na tip detektorja, kjer je bil delec zaznan.

Za boljšo predstavo razporeditve položajev detekcij izrisujemo tiste, ki so

bližje kameri z večjimi točkami. Primer dinamične spremembe velikosti trkov glede na oddaljenost od kamere si lahko ogledamo na sliki 2.23.

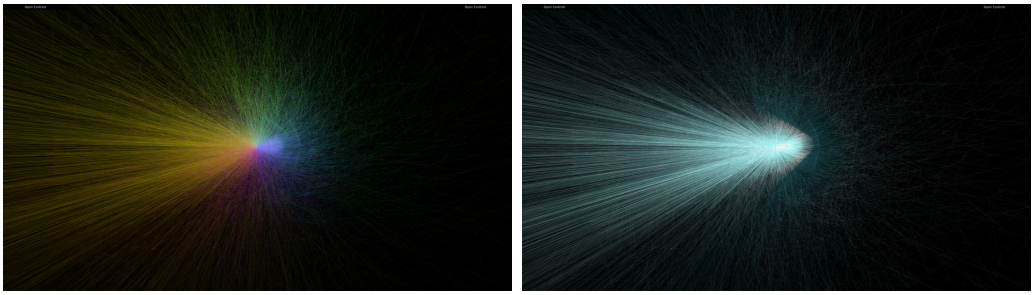


Slika 2.23: Slika prikazuje trke delcev. Na levi strani so trki, kjer njihove velikosti ne spreminjamo ne glede na oddaljenost od kamere. Na desni strani so trki, kjer njihove velikosti spreminjamo glede na oddaljenost od kamere.

Izbiro elementa dogodka v sceni poudarimo s spremembo velikosti točke, ki ponazarja trk, barve in prosojnosti.

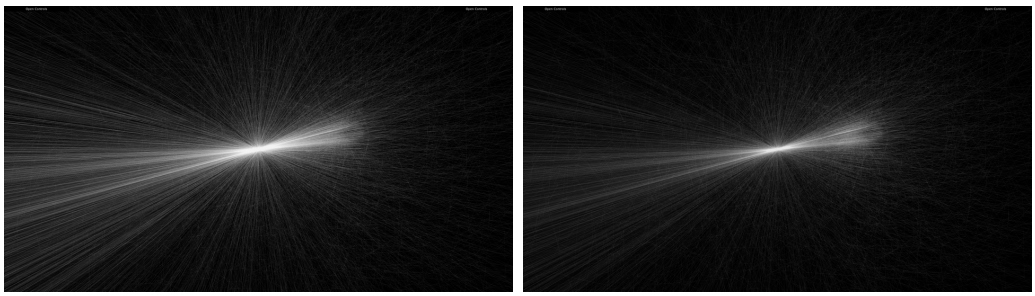
2.5.4 Trajektorije

Senčenje rekonstruiranih trajektorij smo realizirali z nadzorom nad debelino, barvo in prosojnostjo. Barvo določimo glede na atribut gibalne količine v delu trajektorije, glede na tip detektorja v katerem se nahaja trajektorija, ali pa priredimo enotno barvo. Primer izračuna barve si lahko ogledamo na sliki 2.24.



Slika 2.24: Na levi strani so trajektorije, kjer jim določimo barvo glede na atribut gibalne količine. Na desni strani so trajektorije, kjer jim barvo določimo glede na tip detektorja, skozi katerega potujejo.

Zaradi velikega števila trajektorij smo pri senčenju trajektorij barvo in prosojnost dodatno utežili glede na oddaljenostjo od kamere. Ta pristop omogoča Primer dinamične spremembe prosojnosti trajektorij glede na oddaljenost od kamere si lahko ogledamo na sliki 2.25.



Slika 2.25: Slika prikazuje trajektorije. Na levi strani so trajektorije, kjer njihova prosojnost ne spremeni. Na desni strani so trajektorije, kjer njihova prosojnost spremeni, glede na oddaljenost od kamere.

Pri izbiri trajektorije za opazovanje le-to poudarimo s spremembo debeline, barve in prosojnostjo elementov v sceni.

2.5.5 Izbrane trajektorije

Ker želimo izbiro trajektorije dodatno poudariti, smo se na tem mestu odločili implementirati nadzor nad debelino. Pri spremembi debeline trajektorije smo testirali naslednje rešitve:

- metodo WebGL API-ja `glLineWidth`,
- enakomerno porazdeljene točke in
- upodobitev kot trikotni trak.

Metoda `glLineWidth`

Z metodo WebGL API-ja `glLineWidth` določimo širino rasteriziranih linij. Ta pristop ni podprt na vseh platformah in ne bi zadostili cilju platformne ne-

odvisnosti. Uporabnikom, ki uporabljajo ANGLE[7] za interpretacijo WebGL klicev (npr. v okolju Windows), bi linije obdržale enako debelino.

Enakomerno porazdeljene točke

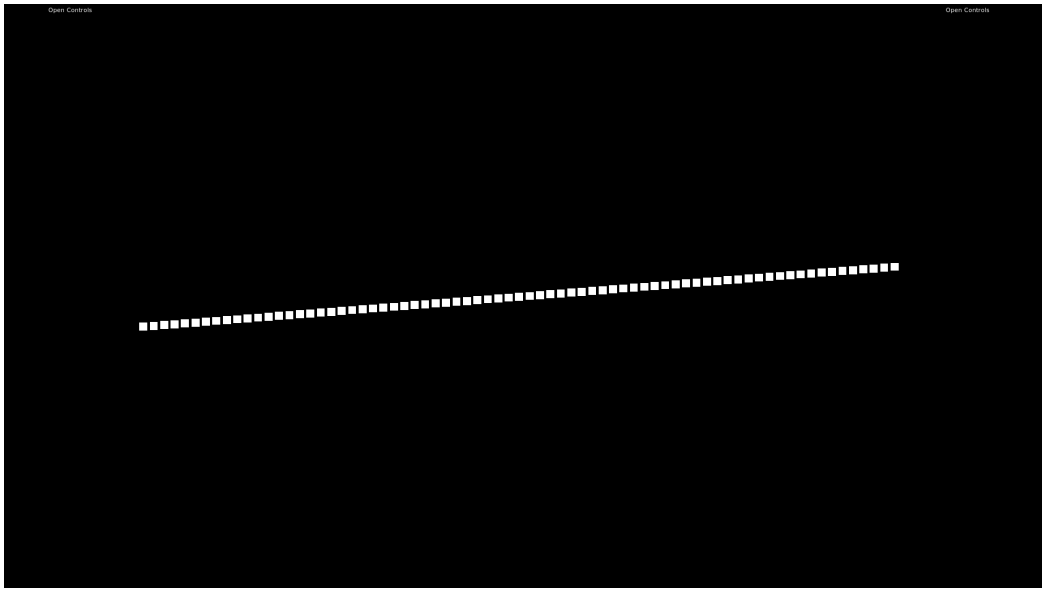
S tem pristopom poskušamo linijo simulirati z enakomerno porazdeljenimi točkami. Pri tem pristopu izračunamo položaje točk za zapolnitev prostora med dvema vozliščema glede na velikost točke in dolžino posameznega segmenta trajektorije. S tem ponazorimo linijo. Izračun položajev je določen s formulo (2.10). Pri tem dx predstavlja širino posamezne točke.

$$\begin{aligned} \boldsymbol{p} &= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} * k * dx \\ \boldsymbol{t} &= \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} \\ k &\in 0, ..., N; N = \frac{\|\boldsymbol{t}\|}{dx} \end{aligned} \tag{2.10}$$

Dobljene točke pošljemo v senčilnik, nad katerimi imamo podprt nadzor nad velikostjo.

To se je izkazalo za izvedljiv pristop, vendar pa take linije niso zvezne v vseh primerih. Do tega pride, ker vmesne položaje poračunamo enkrat na strani glavne procesne enote za večjo hitrost delovanja. Problem predstavlja razdalja linije, ki se ob spreminjanju kamere lahko spremeni, kar lahko privede do primanjkljaja ustvarjenih točk. Primer uporabe pristopa enakomerno porazdeljenih točk si lahko ogledamo na sliki 2.26.

[7] `https://github.com/google/angle`



Slika 2.26: Upodobitev izbrane trajektorije z metodo enakomerno porazdeljenih točk.

Trak trikotnikov

S tem pristopom krajišči vsakega segmenta trajektorije razdelimo na pare točk in vsaki točki v paru dodelimo nasprotno smer. Točke nato (v senčilniku) pravokotno odmaknemo v smeri normale, da pridemo do želene debeline (slika 2.27). Pri tem upoštevamo položaj kamere in dimenzije platna. Dobljene trakove nato izrišemo (s primitivom gl.TRIANGLE_STRIP). To je tudi končna uporabljena rešitev (slika 2.28).

Izračun debeline segmenta trajektorije v senčilniku je ponazorjen v spodnjem segmentu.

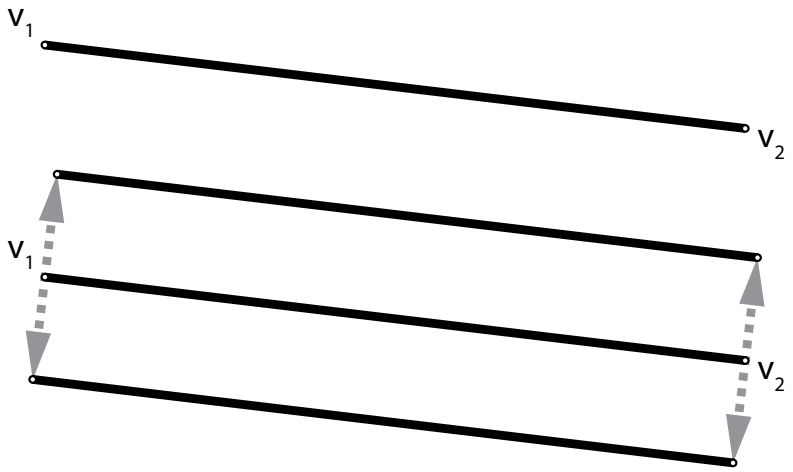
```
//izračun normale daljice
vec2 direction = normalize(nextVertex - prevVertex);
vec2 normal = vec2(-direction.y, direction.x);

//odmik dolžine normale
```

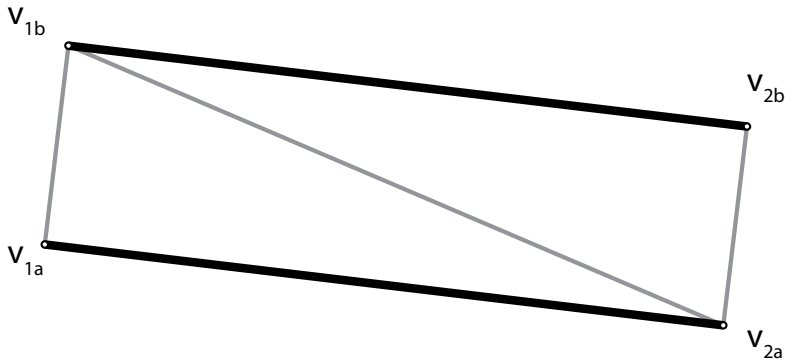
```
normal = normal * line_width / 2.0;
normal.x = normal.x / aspect;

//odmik točke v smeri normale (-1, 1) za njeno dolžino
vec4 offset = vec4(normal * normalDirection, 0.0, 1.0);
gl_Position = currentVertex + offset;
```

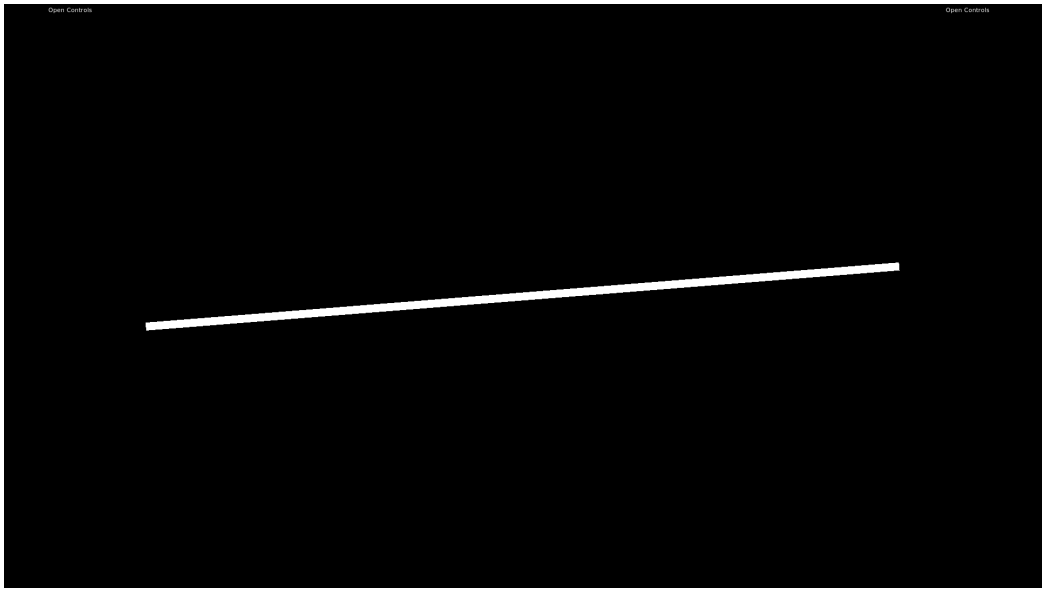
Primer uporabe pristopa traku trikotnikov si lahko ogledamo na sliki 2.29.



Slika 2.27: Ponazoritev odmika v smeri normale za posamezni segment trajektorije.



Slika 2.28: Ponazoritev izrisa trikotnikov za posamezni segment trajektorije.



Slika 2.29: Upodobitev izbrane trajektorije z metodo traka trikotnikov.

2.6 Scena

Z uporabo knjižnice Three.js si pomagamo pri inicializaciji scene, ki jo povežemo z WebGL upodabljevalnikom. V sceni definiramo objekte, kot so luči, navidezna kamera in geometrija objektov eksperimenta. Sceni dodamo nadzor nad kamero z miško in tipkovnico.

Za namene realizacije intuitivne vizualizacije smo za prikaz celotnega eksperimenta upodobili sceno, ki zajema sledeče objekte:

- navidezno kamero,
- luči in
- objekte eksperimenta:
  - geometrijo eksperimenta z detektorji,
  - začetne položaje delcev,
  - zaznane trke delcev z detektorji in

– rekonstruirane trajektorije.

2.6.1 Navidezna kamera

Po sceni navigiramo s prosto navidezno kamero z uporabo miške in tipkovnice. Navidezna kamera, ki jo uporabljamo, je perspektivnega tipa. Kamera je usmerjena v izhodišče sistema, okoli katerega lahko krožimo. S kamero se lahko približamo ali pa oddaljimo od zajetega dogodka trka delcev. Obenem se lahko pomikamo tudi horizontalno in vertikalno. Kamera igra pomembno vlogo pri izbiri elementov za opazovanje, kot so trki in trajektorije. Ta se ob izbiri elementa postopoma pomakne proti izbranemu elementu, v katerega se usmeri in vanj osredotoči pogled.

2.6.2 Luči

V sceni smo se odločili uporabiti 6 točkastih luči (angl. point light), ki smo jih prerazporedili na ploskve navidezne omejevalne koce, ki obsega celoten dogodek eksperimenta. Lučem smo priredili belo barvo in določili jakost sevanja, ki ne pada z razdaljo. S tem smo dosegli, da so objekti v sceni nazorno osvetljeni.

2.6.3 Objekti eksperimenta

V sceni imamo sledeče objekte, ki opredeljujejo fizikalni sistem eksperimenta trka delcev:

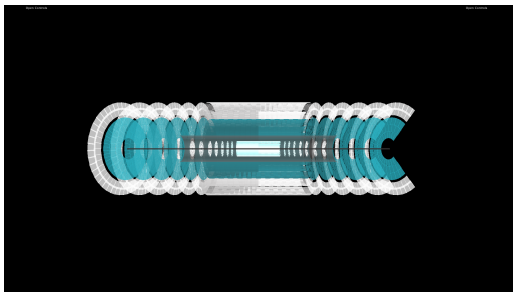
- geometrijo eksperimenta z detektorji,
- začetne položaje delcev,
- zaznane trke delcev z detektorji in
- rekonstruirane trajektorije.

Detektorji

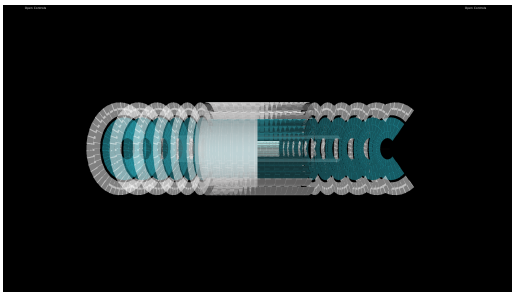
Geometrija detektorjev predstavlja skupino poddetektorjev, ki so prilagojeni zaznavanju določenega tipa delca ali določene lastnosti delcev. V sceni lahko posamezne detektorje prikazujemo ali zakrivamo neodvisno od ostalih. Vsakemu lahko določamo stopnjo prosojnosti in barvo.

Pri realizaciji prosojnosti detektorjev smo prišli na zamisel, ki dobro simulira pravilno dvostransko prosojnost. Ker se posamezni detektorji v prostoru prekrivajo in imajo cilindrično obliko, prosojnost simuliramo tako, da jih najprej izrisujemo od zunanjih proti notranjim brez testa globine. S tem omogočimo prosojen pogled skozi detektorje iz središča navzven. Ker smo test globine pri tem onemogočili, detektorji ostanejo izrisani v obratnem vrstnem redu (slika 2.30). Zato izrišemo v drugem koraku detektorje obratno (od notranjega proti zunanjemu) s testom globine (slika 2.31). S tem omogočimo prosojen pogled v notranjost, zagotovimo pa pravilen izris geometrije glede na globino.

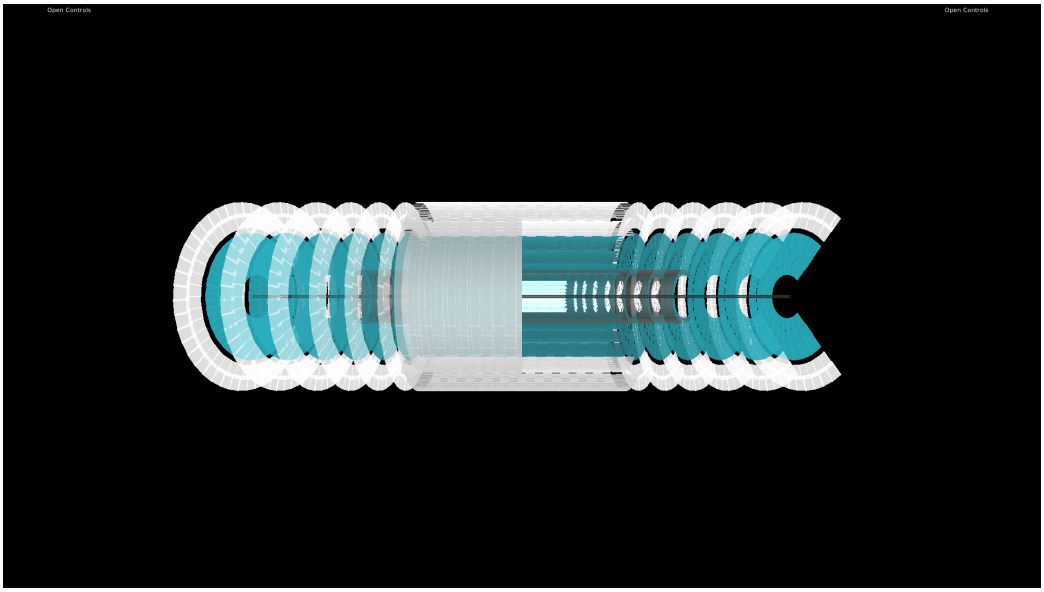
S kombinacijo teh dveh pristopov lahko nazorno simuliramo prosojnost v notranjost detektorjev in navzven (slika 2.32). Z združenjem teh dveh pristopov



Slika 2.30: Detektorji so izrisani od najbolj zunanjega do notranjega.



Slika 2.31: Detektorji so izrisani od notranjega proti zunanjemu.



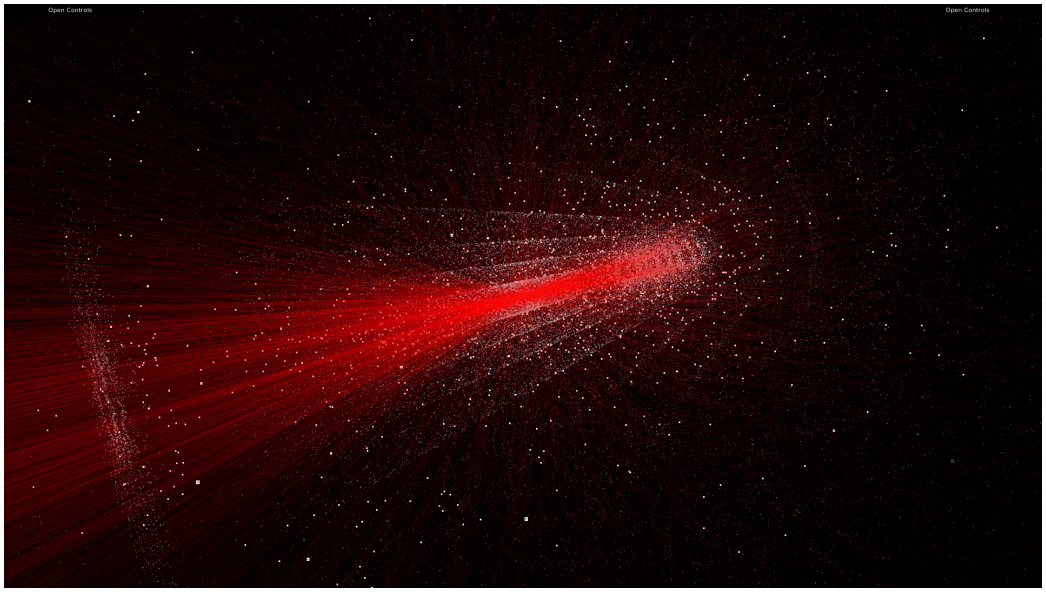
Slika 2.32: Primer upodobitve detektorjev. Detektorji sp najprej izrisani od najbolj zunanjega do notranjega in nato od notranjega proti zunanjemu.

Ta pristop omogoča, da le z dodatnim klici za izris vsakega poddetektorja in brez urejanja primitivov učinkovito simuliramo obojestransko prosojnost vseh objektov v sceni.

Delci, trki in trajektorije

Geometrija delcev predstavlja točke položajev, kjer so bili delci v začetem položaju v času trka. Delcem v začetnih položajih lahko spreminjamo vidljivost, barvo, prosojnost in način določevanja barve. Geometrija trkov predstavlja točke položajev, kjer so bili zaznani trki delcev z poddetektorji. Trkom lahko določimo vidljivost, spreminjamo barvo in stopnjo prosojnosti. Lahko jim določimo tudi način določanja barve. Geometrija trajektorij predstavlja položaje delcev in trkov, ki se skupaj povezujejo v trajektorije. trajektorijam lahko določimo vidljivost, spreminjamo barvo in stopnjo prosojnosti. Trajektorijam lahko tudi določimo način določanja barve.

Slika 2.33 predstavlja množico upodobljenih elementov eksperimenta.



Slika 2.33: Primer upodobitve delcev, trkov in trajektorij.

2.7 Uporabniški vmesnik

Za lažjo interakcijo smo implementirali uporabniški vmesnik na osnovi JavaScript knjižnice dat.GUI.

Ob zagonu sistema se v sceni prikaže geometrija detektorja in uporabniški vmesnik. Po sceni se lahko premikamo z orbitalno kamero z uporabo miške in tipkovnice. Uporabniški vmesnik nam ponuja nastavitve posameznih atributov (rezalne ravnine, vidljivost, barva, prosojnost, velikost, tip obarvanja), prikaza detektorjev, delcev, njihovih trkov z detektorji in rekonstruiranih trajektorij. Posamezni dogodek za opazovanje lahko izberemo iz prikazanega seznama, ki nam ob izbiri prikaže postopek nalaganja. Ob izbiri dogodka se v sceno naloži delce, trke in trajektorije.

Ker je elementov veliko, smo implementirali možnost iskanja preko identifikacijske številke. Za opazovanje lahko izberemo določen trk ali trajektorijo delca, pri čemer se kamera postavi ob izbrani element, izbiro pa še dodatno ponazorimo z obarvano označbo v seznamu, ki pripada prirejeni barvi ele-

menta. V primeru izbire trajektorije se kamera postopoma pomakne med začetno in končno vozlišče trajektorije (odmaknjeno v smeri normale) ter se usmeri v njen centroid. Ob izbiri trka se kamera postopoma pomakne med središče sistema in trk ter se vanj usmeri.

Z namenom pomoči tekmovalcem CERN-ovega izziva TrackML smo dodali tudi možnost nalaganja lastnega dogodka eksperimenta, ki ga lahko primerjamo s katerim izmed izbranih. Na tem mestu smo implementirali funkcionalnost, ki ob izbiri določene trajektorije priredi seznam najbolj podobnih trajektorij glede na centroid trajektorije in število njenih segmentov.

Uporabniški vmesni zajema sledeče komponenete:

- vmesnik dogodkov,
- vmesnik trkov,
- vmesnik trajektorij in
- vmesnik parametrov.

2.7.1 Vmesnik dogodkov

Uporabniški vmesnik dogodkov se nahaja na levi strani. Vsebuje seznam dogodkov, po katerem lahko navigiramo s prirejenima gumboma in priloženim poljem za direktno iskanje. Polje za iskanje smo realizirali tako, da se ob vnosu posameznega znaka izvede iskanje po vseh dogodkih. Rezultat iskanja je seznam dogodkov, ki vsebujejo vneseni niz v identifikacijskem številu.

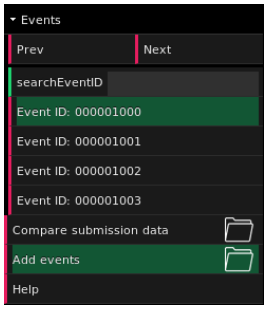
Vmesnik dogodkov vsebuje tudi možnost, da dodamo lastne in poljubne dogodke v seznam. To naredimo na tak način, da preko izbirnika datotek (angl. file chooser) uporabnik v sistem doda datotečne vpisnike (angl. file descriptor) datotek, ki opisujejo dogodek. S tem se izognemo, da ne bi v pomnilnik po nepotrebnem nalagali velike količine podatkov, ampak to storimo, zgolj ko uporabnik izbere določen dogodek s seznama.

Ob izbiri dogodka preverimo za napake ob branju ali prenosu datoteke. Potek prenosa ponazorimo z postopkom nalaganja, ki smo ga dodali v vme-

snik. Ob uspešno izbranem dogodku, to ponazorimo z obarvanjem označbe in omogočimo primerjavo lastnega dogodka. Na tem mestu dodamo tudi vmesnik s seznamom trkov in trajektorij, v sceno pa naložimo geometrijo izbranega dogodka eksperimenta.

V primeru, da uporabnik želi primerjati rezultate lastno rekonstruiranih trajektorij z dogodkom zlatega standarda, se pojavi dodatni vmesnik trajektorij uporabnika.

Primer vmesnika dogodkov in izbranega dogodka si lahko ogledamo na sliki 2.34;



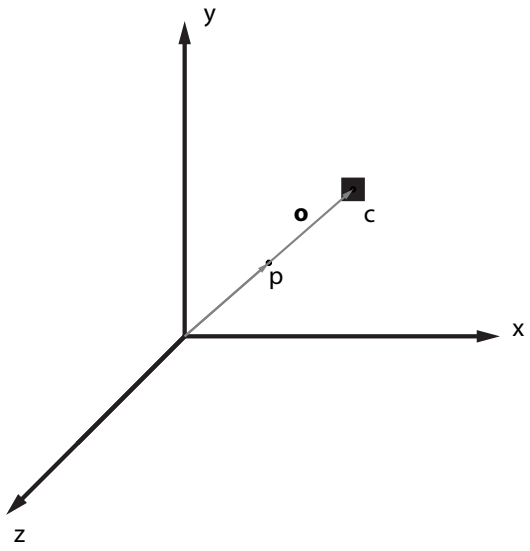
Slika 2.34: Ponazoritev vmesnika dogodkov, kjer smo izbrali dogodek.

2.7.2 Vmesnik trkov

Uporabniški vmesnik trkov se pojavi na desni strani ob izbiri določenega dogodka s seznama. Vsebuje seznam vseh trkov delcev fizikalnega eksperimenta izbranega dogodka. Po seznamu lahko navigiramo z uporabo gumbov in poljem za direktno iskanje. Polje za iskanje smo realizirali tako, da se ob vnosu posameznega znaka izvede iskanje po vseh trkih. Rezultat iskanja je seznam trkov, ki vsebujejo vneseni niz v identifikacijskem številu.

Ob izbiri določenega trka to poudarimo z obarvano označbo v seznamu, ki pripada prirejeni barvi upodobljenega trka v sceni. Na tem mestu se kamera postavi ob izbrani element. Kamera se postopoma pomakne med središče sistema in trk ter se vanj usmeri.

Shemo izračuna položaja (p) in orientacije ($\boldsymbol{o}$) kamere si lahko ogledamo na sliki 2.35.



Slika 2.35: Slika prikazuje shemo izbire položaja (p) in orientacije ($\boldsymbol{o}$) kamere ob izbiri trka.

Položaj izračunamo po formuli (2.11) in orientacijo po formuli (2.12).

$$p = a * c; a \in \mathbb{R} \tag{2.11}$$

$$\boldsymbol{o} = c \tag{2.12}$$

Primer vmesnika trkov izbranega dogodka si lahko ogledamo na sliki 2.36;

2.7.3 Vmesnik trajektorij

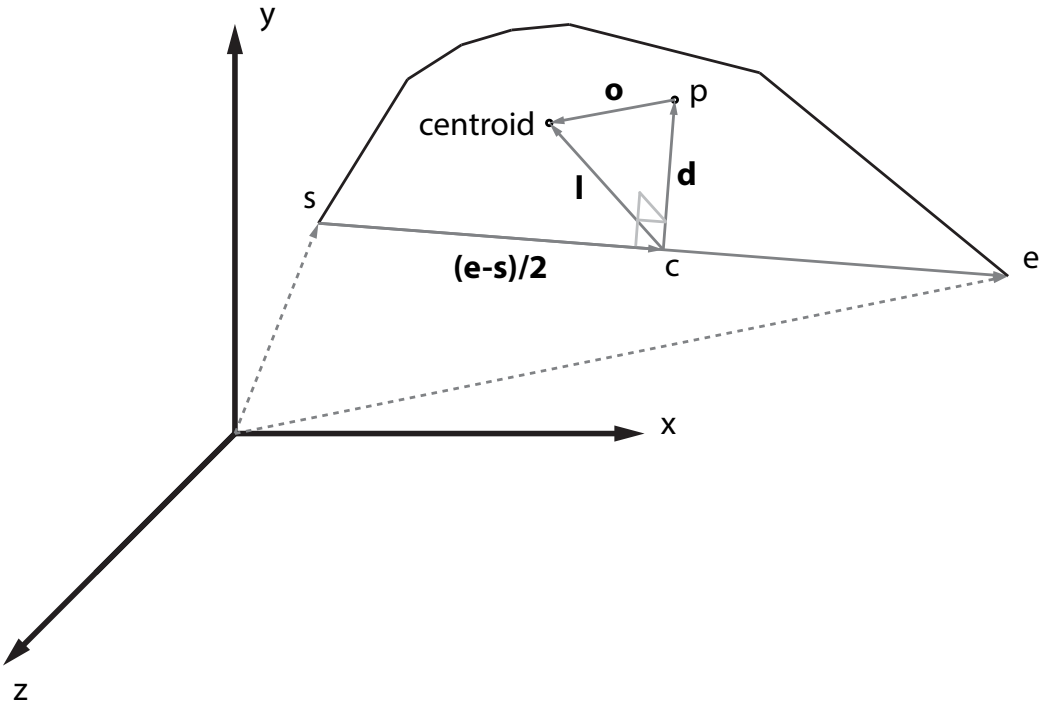
Uporabniški vmesnik trajektorij se pojavi na desni strani ob izbiri dogodka s seznama. Vsebuje seznam vseh trajektorij delcev fizikalnega eksperimenta izbranega dogodka. Po seznamu lahko navigiramo z uporabo gumbov in poljem za direktno iskanje. Polje za iskanje smo realizirali tako, da se ob vnosu posameznega znaka izvede iskanje po vseh trajektorijah. Rezultat

Slika 2.36: Ponazoritev vmesnika trkov, kjer smo izbrali trk.

iskanja je seznam trajektorij, ki vsebujejo vneseni niz v identifikacijskem številu.

Ob izbiri določene trajektorije to poudarimo z obarvano označbo v seznamu, ki pripada prirejeni barvi upodobljene trajektorije v sceni. Na tem mestu se kamera postavi ob izbrani element. Trajektorije so različnih oblik in različnih dolžin v prostoru $\mathbb{R}^3$. Poleg tega so same trajektorije zgrajene iz različnega števila segmentov daljic, ki so prav tako lahko različnih dolžin. To lahko privede do slabe in nejasne izbire položaja in orientacije kamere v nekaterih primerih trajektorij. Zato smo si morali zamisli konsistentno primeren način izbire položaja in orientacije kamere ob izbiri trajektorije za ogled. Dobro rešitev smo dosegli tako, da se kamera postopoma pomakne med začetno in končno vozlišče trajektorije (odmaknjeno v smeri normale) ter se usmeri v njen centroid.

Shemo izračuna položaja (p) in orientacije ($\boldsymbol{o}$) kamere si lahko ogledamo na sliki 2.37.



Slika 2.37: Slika prikazuje shemo izbire položaja (p) in orientacije ($\boldsymbol{o}$) kamere ob izbiri trajektorije.

Položaj izračunamo po formuli (2.13) in orientacijo po formuli (2.14).

$$p = c + a * \boldsymbol{d}; a \in \mathbb{R} \tag{2.13}$$

$$\boldsymbol{o} = centroid - p \tag{2.14}$$

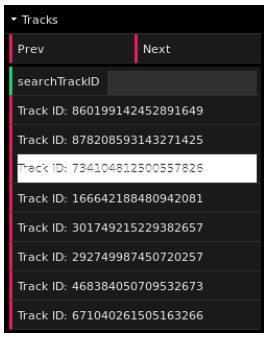
$$c = s + \frac{\boldsymbol{e} - \boldsymbol{s}}{\boldsymbol{2}}$$

$$\boldsymbol{l} = centroid - c$$

$$\boldsymbol{d} = \boldsymbol{l} \times \frac{\boldsymbol{e} - \boldsymbol{s}}{\boldsymbol{2}}$$

V primeru, da uporabnik primerja rezultate lastno zgrajenih trajektorij, se ob izbiri določene trajektorije zlatega standarda uredi seznam uporabnikovih trajektorij. Urejanje smo implementirali na način, ki primerja centroide trajektorij in število vozlišč, ki sestavlja trajektorijo. Trajektorije se nato uredijo od najbolj do najmanj podobne izbrani.

Primer vmesnika trajektorij izbranega dogodka si lahko ogledamo na sliki 2.38;



Slika 2.38: Ponazoritev vmesnika trajektorij, kjer smo izbrali trajektorijo.

2.7.4 Vmesnik parametrov

Uporabniški vmesnik parametrov se nahaja na desni strani. Vsebuje parametre geometrije detektorjev in parametre lastnosti delcev, njihovih trkov in rekonstruiranih trajektorij.

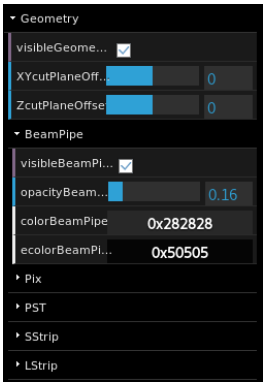
Detektorji

S parametri geometrije lahko določamo rezalne ravnine in vidljivost celotne geometrije detektorja. S tem si lahko pomagamo za boljši pogled v notranjost detektorja.

Določimo lahko tudi parametre posameznih detektorjev. Na primer vidljivost posameznega detektorja, njegovo stopnjo prosojnosti in barve za izračun osvetlitve. Slika 2.39 predstavlja vmesnik parametrov, ki jih lahko določimo za vsak detektor.

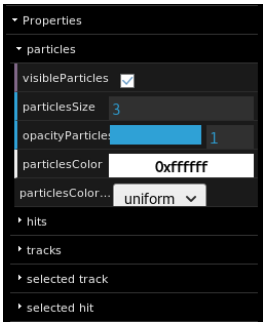
Lastnosti

Posamezne lastnosti lahko določamo delcem, trkom, trajektorijam, izbranim trkom in izbranim trajektorijam. Določamo lahko lastnosti, kot so vidlji-



Slika 2.39: Vmesnik parametrov, ki jih lahko nastavimo za vsak detektor.

vost, barva in prosojnost. Določimo lahko tudi velikost in način obarvanja elementov. Slika 2.40 predstavlja vmesnik parametrov, ki jih lahko določimo za vsak element znotraj dogodka eksperimenta.

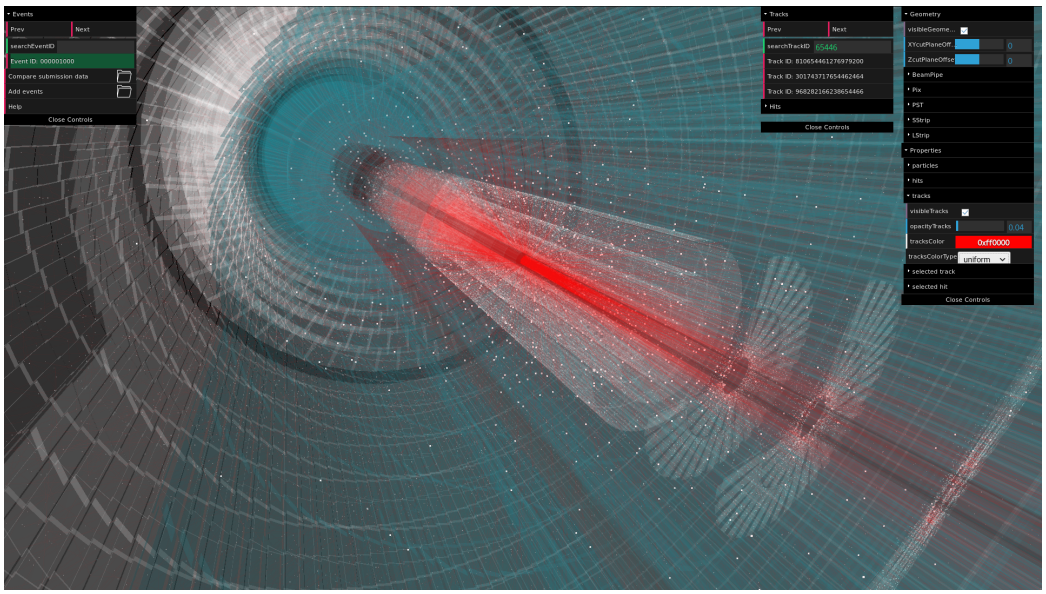


Slika 2.40: Vmesnik prametrov, ki jih lahko nastavimo za vsak element znotraj dogodka eksperimenta.

Poglavje 3

Testiranje, analiza in ovrednotenje rezultatov

Končni izgled razvitega sistema je prikazan na sliki 3.1. Slika prikazujeta primer upodobitve enega dogodka trka delcev.



Slika 3.1: Primer upodobitve fizikalnega dogodka trka delcev z našim sistemom.

3.1 Specifikacije testnega sistema

Sistem smo testirali na prenosniku s spodnjimi specifikacijami:

- dvojedrni procesor Intel Core i7 4510U pri taktu 2.00 GHz,
- 8 GB glavnega pomnilnika, 8 GB dodatnega pomnilnika na disku (angl. swap),
- grafična kartica AMD Radeon R7 M260.

3.2 Izhodiščni sistem

Izhodiščni sistem predstavlja povzetek obstoječih sistemov, ki se uporabljajo v znanosti fizike osnovnih delcev pri visokih energijah. Izhodiščni sistem ni optimiziran in počasi deluje na številnih platformah. Pri obdelavi dogodka z običajno količino podatkov na naši testni napravi sistem odpove, saj nam zmanjka glavnega in dodatnega pomnilnika na disku. To povzroči, da se sistem kot proces ukine.

Test predstavlja merjenje časa izrisovanja posamezne slike (angl. frame time) pri različnih dimenzijah platna. Da bodo testi enakovredni, v sistemu spremenimo nekaj nastavitev in parametrov. Phongove materiale spremenimo v Lambertove in zmanjšamo število luči na šest. V sistemu tudi izklopimo mehčanje robov (angl. antialiasing). Za kamero izberemo enoten položaj in orientacijo. Velikost problema N predstavlja dimenzijo platna. Dimenzijo platna povečujemo po standardnih dimenzijah razmerja 16:9. Za dimenzije platna izberemo $N = \{854 \times 480, 1024 \times 576, 1280 \times 720, 1366 \times 768, 1600 \times 900, 1920 \times 1080\}$. Za upodabljanje v obeh podtestih izberemo dogodek, ki po velikosti predstavlja podmnožico običajnega dogodka. V prvem podtestu omogočimo prikaz geometrije detektorja, trkov in trajektorij. V drugem podtestu onemogočimo prikaz geometrije detektorja. Omogočimo prikaz trkov in trajektorij. Detektorje v tem podtestu onemogočimo, ker se zaradi prekrivanja detektorjev izloči večina trajektorij in trkov. Ker se

izločijo, niso vidni in se posledično ne upodabljajo zaradi rezanja v grafičnem cevovodu. Pri obeh podtestih merimo čas izrisa slike. Pri vsakem izmed testnih primerov izmerimo 10.000 neodvisnih vzorcev. Dobljene rezultate povprečimo in zanje izračunamo standardni odklon (σ).

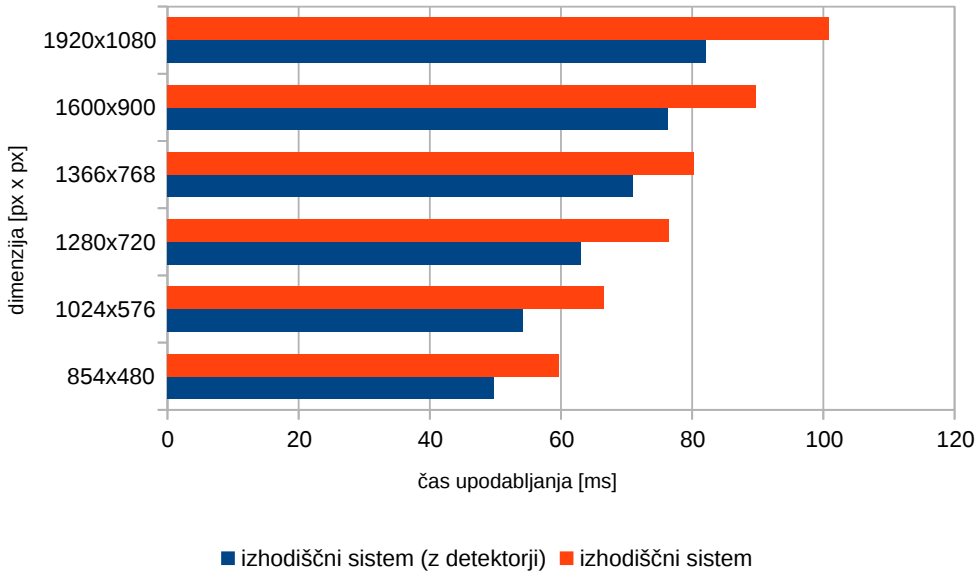
Rezultati merjenja prvega podtesta so v tabeli 3.1, rezultati drugega podtesta pa v tabeli 3.2. Graf rezultatov obeh podtestov je prikazan na sliki 3.2. Časi upodabljanja obeh podtestov se povečujejo skupaj z večanjem dimenzije platna. Najhitreje deluje sistem prvega podtesta, ki ima omogočen prikaz geometrije detektorja. Pri dimenziji platna 854×480 izriše posamezno sliko v 49.76 ms, kar je približno enako 20 slik na sekundo. Razvidno je, da v primeru omogočenega izrisa detektorjev (prvi podtest) sistem deluje hitreje. Razlog tega je rezanje, ki zaradi neprosojnih detektorjev odstrani velik del trkov in trajektorij.

N	**Čas računanja [ms]**	**σ časa računanja [ms]**
854×480	49.76	4.85
1024×576	54.21	5.32
1280×720	62.99	6.76
1366×768	71.01	8.87
1600×900	76.36	8.08
1920×1080	82.07	4.47

Tabela 3.1: Čas računanja ter standardni odklon meritev (σ) v odvisnosti od velikosti problema (N) za prvi podtest.

N	Čas računanja [ms]	σ časa računanja [ms]
854 × 480	59.743	3.889
1024 × 576	66.618	4.359
1280 × 720	76.506	4.307
1366 × 768	80.309	5.028
1600 × 900	89.655	4.135
1920 × 1080	100.863	3.899

Tabela 3.2: Čas računanja ter standardni odklon meritev (σ) v odvisnosti od velikosti problema (N) za drugi podtest.



Slika 3.2: Časi upodabljanja izhodiščnega sistema v odvisnosti od dimenzije platna.

3.3 Izboljšani sistem v Three.js ogrodju

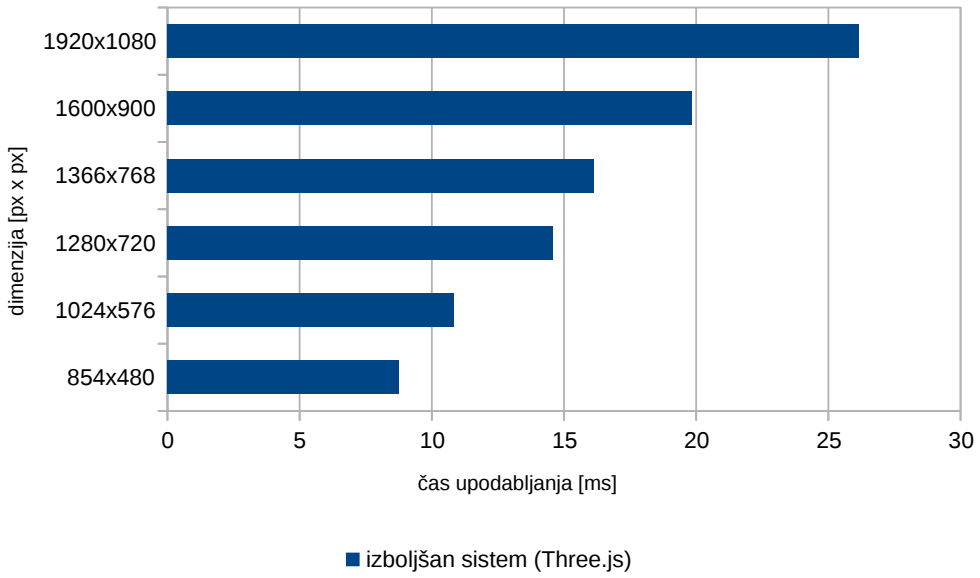
Izboljšani sistem predstavlja rezultat optimizacije in izboljšane vizualizacije fizikalnega eksperimenta trka delcev.

Test predstavlja merjenje časa izrisovanja posamezne slike pri različnih dimenzijah platna. Za kamero izberemo enoten položaj in orientacijo. Velikost problema N predstavlja dimenzijo platna. Dimenzijo platna povečujemo po standardnih dimenzijah razmerja 16:9. Za dimenzije platna izberemo $N = \{854 \times 480, 1024 \times 576, 1280 \times 720, 1366 \times 768, 1600 \times 900, 1920 \times 1080\}$. V testu izberemo enotni dogodek in omogočimo prikaz geometrije detektorja, delcev, trkov in trajektorij. Pri testu merimo čas izrisa slike. Pri vsakem izmed testnih primerov izmerimo 10.000 neodvisnih vzorcev. Dobljene rezultate povprečimo in zanje izračunamo standardni odklon (σ).

Rezultati merjenja testa so v tabeli 3.3. Graf rezultatov testa je prikazan na sliki 3.3. Časi upodabljanja se povečujejo skupaj z večanjem dimenzije platna. Pri dimenziji platna 854×480 izriše posamezno sliko v 8.747 ms, kar je približno enako 114 slik na sekundo.

N	**Čas računanja [ms]**	**σ časa računanja [ms]**
854 × 480	8.747	1.211
1024 × 576	10.836	1.227
1280 × 720	14.576	1.165
1366 × 768	16.129	1.255
1600 × 900	19.843	1.317
1920 × 1080	26.139	1.446

Tabela 3.3: Čas računanja ter standardni odklon meritev (σ) v odvisnosti od velikosti problema (N).



Slika 3.3: Časi upodabljanja optimiziranega sistema v odvisnosti od dimenzije platna.

3.3.1 Indeksiranje trajektorij

Indeksiranje trajektorij predstavlja dodatno optimizacijo v izboljšanem sistemu.

Test predstavlja merjenje časa izrisovanja posamezne slike pri različnem številu trajektorij v sceni. Tu smo merili doprinos uporabe indeksirane geometrije trajektorij (izboljšanega sistema) v odvisnosti od njihovega števila. Za kamero izberemo enoten položaj in orientacijo. Velikost problema N predstavlja število trajektorij v sceni. Število trajektorij povečujemo s korakom 10.000. Za število trajektorij izberemo $N = \{10.000, 20.000, 30.000, 40.000, 50.000, 60.000\}$. Platno izrisujemo z dimenzijo 1920×1080. V testu izberemo enotni dogodek in onemogočimo prikaz geometrije detektorja, delcev, trkov. V testu omogočimo zgolj prikaz trajektorij. V prvem podtestu trajektorije upodabljamo brez uporabe indeksiranja. V drugem podtestu trajektorije upodabljamo z uporabo indeksiranja. Pri obeh podtestih merimo čas izrisa

slike. Pri vsakem izmed testnih primerov izmerimo 10.000 neodvisnih vzorcev. Dobljene rezultate povprečimo in zanje izračunamo standardni odklon (σ). Za vsak testni primer izračunamo mero pohitritve S, ponazorjeno s formulo (3.1), kjer T_1 predstavlja čas upodabljanja brez indeksiranja in T_2 čas upodabljanja z indeksiranjem.

$$S = \frac{T_1}{T_2} \tag{3.1}$$

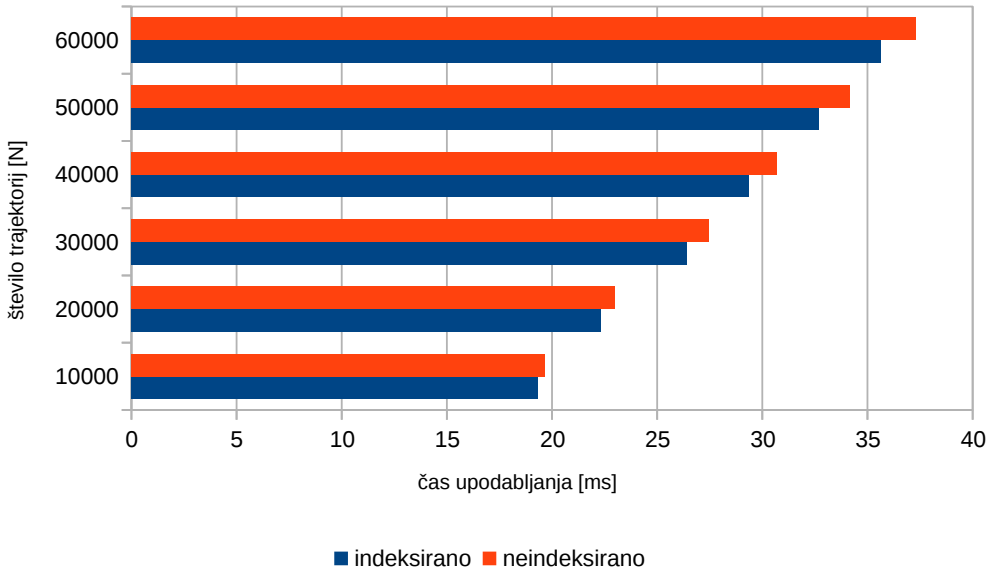
Rezultati merjenja prvega podtesta so v tabeli 3.4 in drugega podtesta v tabeli 3.5. Graf rezultatov testa je prikazan na sliki 3.4. Mera pohitritve S je prikazana na sliki 3.5. Časi upodabljanja se povečujejo skupaj z večanjem števila trajektorij. Razvidno je, da se doprinos veča s številom prisotnih trajektorij, kar je tudi pričakovano. Ker je v dogodku trka delcev običajno nekje med 10.000 in 20.000 trajektorij, je glede na rezultate pričakovan doprinos indeksiranja nekje med 2 in 3 odstotke.

N	**Čas računanja [ms]**	**σ časa računanja [ms]**
10.000	19.681	1.734
20.000	23.008	1.726
30.000	27.456	1.413
40.000	30.689	1.375
50.000	34.184	1.632
60.000	37.320	1.535

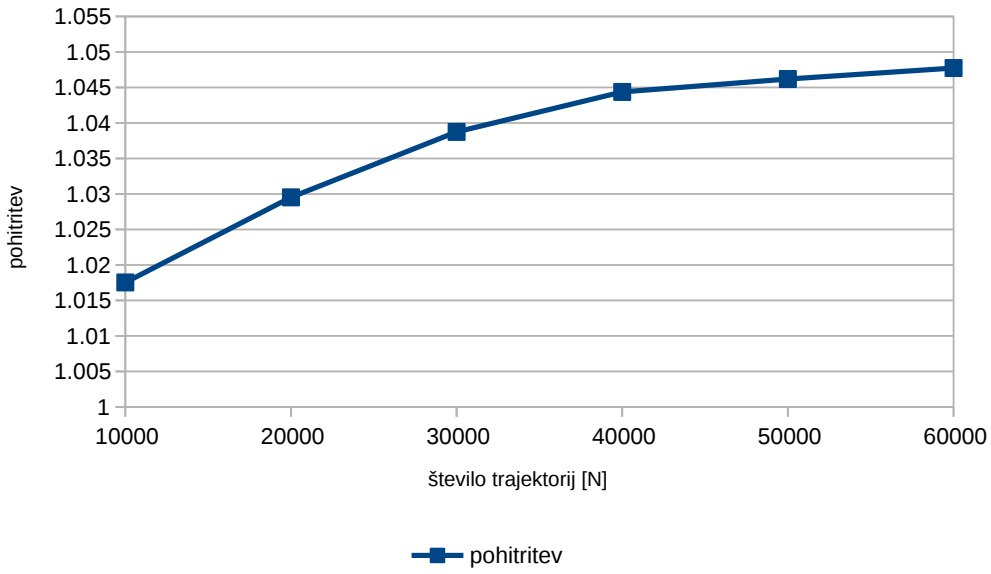
Tabela 3.4: Čas računanja ter standardni odklon meritev (σ) v odvisnosti od velikosti problema (N) za prvi podtest.

N	Čas računanja [ms]	σ časa [ms]	Pohitritev (S)
10.000	19.342	1.398	1.018
20.000	22.348	1.432	1.029
30.000	26.432	1.323	1.039
40.000	29.685	1.441	1.044
50.000	32.675	1.305	1.046
60.000	35.619	1.511	1.048

Tabela 3.5: Čas računanja, standardni odklon meritev (σ) in pohitritev (S) v odvisnosti od velikosti problema (N) za drugi podtest.



Slika 3.4: Čas upodabljanja trajektorij v odvisnosti od njihovega števila.



Slika 3.5: Pohitritev sistema v odvisnosti od števila trajektorij.

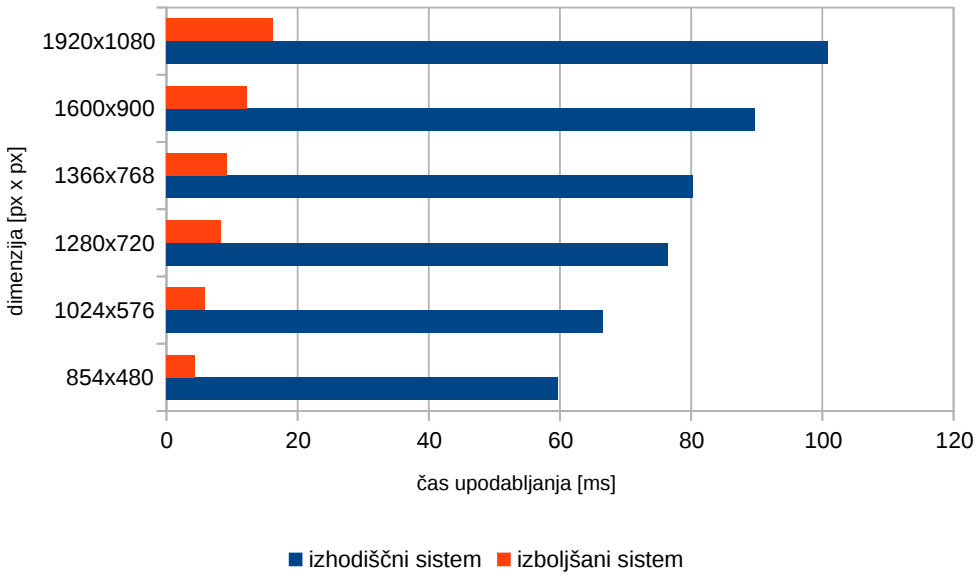
3.3.2 Pohitritev sistema

Testirali smo pohitritev izboljšanega sistema v primerjavi z izhodiščnim sistemom. Primerjali smo čas upodabljanja obeh sistemov v odvisnosti od dimenzije platna. Za kamero izberemo enoten položaj in orientacijo. Velikost problema N predstavlja dimenzijo platna. Dimenzijo platna povečujemo po standardnih dimenzijah razmerja 16:9. Za dimenzije platna izberemo $N = \{854 \times 480, 1024 \times 576, 1280 \times 720, 1366 \times 768, 1600 \times 900, 1920 \times 1080\}$. Za upodabljanje v testu izberemo dogodek, ki po velikosti predstavlja podmnožico običajnega dogodka. Omogočimo prikaz trkov in trajektorij. S temi nastavitvami se najbolje prilegamo izhodiščnemu sistemu. Pri testu merimo čas izrisa slike. Pri vsakem izmed testnih primerov izmerimo 10.000 neodvisnih vzorcev. Dobljene rezultate povprečimo in zanje izračunamo standardni odklon (σ). Za vsak testni primer izračunamo mero pohitritve S, ponazorjeno s formulo (3.1), kjer T_1 predstavlja čas upodabljanja na izhodiščnem sistemu in T_2 čas upodabljanja na izboljšanem sistemu.

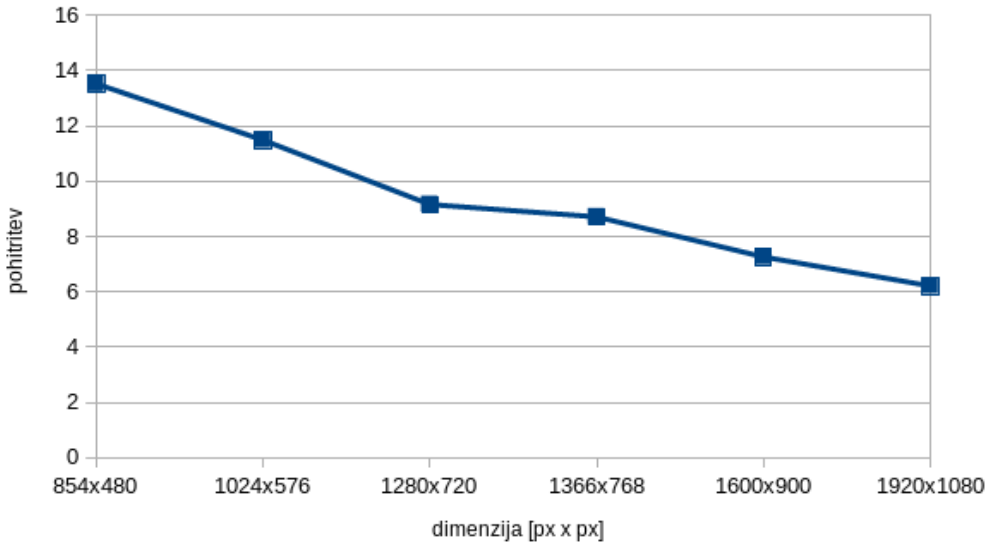
Rezultati merjenja izhodiščnega sistema so v tabeli 3.2. Rezultati merjenja testa pohitritve izboljšanega sistema so v tabeli 3.6. Graf rezultatov testa je prikazan na sliki 3.6. Mera pohitritve S je prikazana na sliki 3.7. Časi upodabljanja se povečujejo skupaj z večanjem dimenzije platna. Izboljšani sistem deluje hitreje kot izhodiščni pri vseh testnih primerih. Z večanjem dimenzije platna opazimo, da se pohitritev manjša. Pri dimenziji platna 1920×1080 znaša pohitritev približno 6.

N	**Čas računanja [ms]**	**σ časa [ms]**	**Pohitritev (S)**
854×480	4.417	0.912	13.525
1024×576	5.798	1.057	11.489
1280×720	8.352	1.141	9.159
1366×768	9.218	1.181	8.712
1600×900	12.343	1.598	7.264
1920×1080	16.240	1.439	6.211

Tabela 3.6: Čas računanja, standardni odklon meritev (σ) in pohitritev (S) v odvisnosti od velikosti problema (N).



Slika 3.6: Čas upodabljanja izhodiščnega in izboljšanega sistema v odvisnosti od dimenzije platna.



Slika 3.7: Pohitritev sistema v odvisnosti od dimenzije platna.

3.4 Izboljšani sistem v Med3D ogrodju

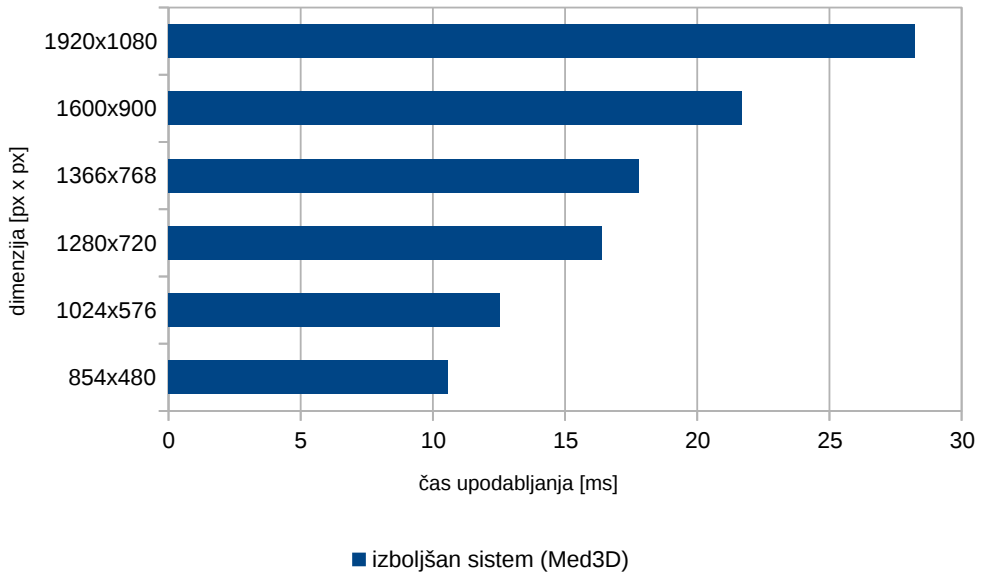
V ogrodje Med3D smo vgradili logiko in funkcionalnost izboljšanega sistema.

Test predstavlja merjenje časa izrisovanja posamezne slike v odvisnosti od dimenzije platna. Za kamero izberemo enoten položaj in orientacijo. Velikost problema N predstavlja dimenzijo platna. Dimenzijo platna povečujemo po standardnih dimenzijah razmerja 16:9. Za dimenzije platna izberemo $N = \{854\times480, 1024\times576, 1280\times720, 1366\times768, 1600\times900, 1920\times1080\}$. V testu izberemo enotni dogodek in omogočimo prikaz geometrije detektorja, delcev, trkov in trajektorij. Pri testu merimo čas izrisa slike. Pri vsakem izmed testnih primerov izmerimo 10.000 neodvisnih vzorcev. Dobljene rezultate povprečimo in zanje izračunamo standardni odklon (σ).

Rezultati merjenja testa so v tabeli 3.7. Graf rezultatov testa je prikazan na sliki 3.8. Časi upodabljanja se povečujejo skupaj z večanjem dimenzije platna.

N	**Čas računanja [ms]**	**σ časa računanja [ms]**
854 × 480	10.574	1.559
1024 × 576	12.533	1.751
1280 × 720	16.415	1.852
1366 × 768	17.791	1.822
1600 × 900	21.679	1.731
1920 × 1080	28.229	1.858

Tabela 3.7: Čas računanja ter standardni odklon meritev (σ) v odvisnosti od velikosti problema (N).



Slika 3.8: Časi upodabljanja sistema v ogrodju Med3D v odvisnosti od dimenzije platna.

3.4.1 Pohitritev sistema

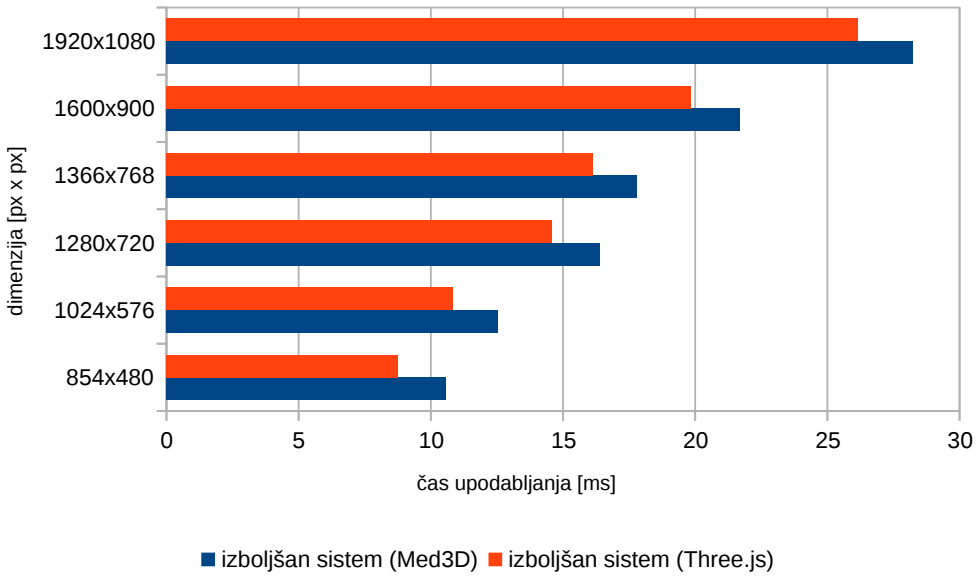
Testirali smo pohitritev izboljšanega sistema z uporabo Three.js v primerjavi z izboljšanim sistemom, ki deluje na Med3D ogrodju. Primerjali smo čas upodabljanja obeh sistemov v odvisnosti od dimenzije platna. Za kamero izberemo enoten položaj in orientacijo. Velikost problema N predstavlja dimenzijo platna. Dimenzijo platna povečujemo po standardnih dimenzijah razmerja 16:9. Za dimenzije platna izberemo $N = \{854 \times 480, 1024 \times 576, 1280 \times 720, 1366 \times 768, 1600 \times 900, 1920 \times 1080\}$. V testu izberemo enotni dogodek in omogočimo prikaz geometrije detektorja, delcev, trkov in trajektorij. Pri testu merimo čas izrisa slike. Pri vsakem izmed testnih primerov izmerimo 10.000 neodvisnih vzorcev. Dobljene rezultate povprečimo in zanje izračunamo standardni odklon (σ). Za vsak testni primer izračunamo mero pohitritve S, ponazorjeno s formulo (3.1), kjer T_1 predstavlja čas upodabljanja na sistemu z uporabo Three.js in T_2 čas upodabljanja na sistemu,

ki deluje na Med3D ogrodju.

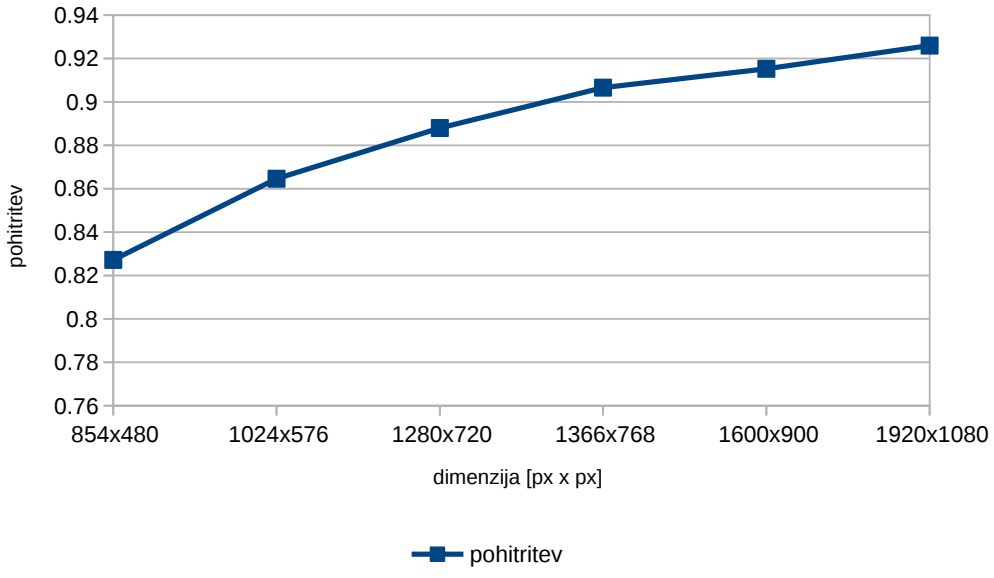
Rezultati merjenja izboljšanega sistema so v tabeli 3.3. Rezultati merjenja testa so v tabeli 3.8. Graf rezultatov testa je prikazan na sliki 3.9. Mera pohitritve S je prikazana na sliki 3.10. Časi upodabljanja se povečujejo skupaj z večanjem dimenzije platna. Izboljšani sistem z uporabo Three.js deluje hitreje kot izboljšani sistem na Med3D ogrodju pri vseh testnih primerih. Z večanjem dimenzije se čas upodabljanja izboljšanega sistema na Med3D ogrodju približuje izboljšanemu sistemu, ku uporablja Three.js. Sistem z uporabo Three.js je v tem primeru hitrejši, ker s trenutno sestavo geometrije ne izkoriščamo prednosti odloženega upodabljanja ogrodja Med3D. Ogrodje Med3D je prilagojeno odloženemu upodabljanju in potrebuje posledično narediti dva izrisa, kar zahteva nekaj dodatnega časa. Pohitritev sistema v ogrodju Med3D bi se najverjetneje povečala z uporabo večjega števila luči in pri uporabi post-produkcijskih senčilnih prehodov.

N	**Čas računanja [ms]**	**σ časa [ms]**	**Pohitritev (S)**
854 × 480	10.574	1.559	0.8272
1024 × 576	12.533	1.751	0.865
1280 × 720	16.415	1.852	0.888
1366 × 768	17.791	1.822	0.907
1600 × 900	21.679	1.731	0.915
1920 × 1080	28.229	1.858	0.926

Tabela 3.8: Čas računanja, standardni odklon meritev (σ) in pohitritev (S) v odvisnosti od velikosti problema (N).



Slika 3.9: Čas upodabljanja sistema, ki uporablja Three.js in sistema v Med3D okolju v odvisnosti od dimenzije platna.



Slika 3.10: Pohitritev sistema v odvisnosti od dimenzije platna.

Poglavje 4

Zaključek

V tem delu smo predstavili sistem, ki omogoča interaktivno in intuitivno vizualizacijo trkov osnovnih delcev. Sistem omogoča interakcijo z upodobljenimi podatki in nadzor nad raznimi parametri vizualizacije.

Sistem smo razvili z namenom optimizacije že obstoječih sistemov, ki se uporabljajo v znanosti fizike osnovnih delcev pri visokih energijah. Poleg tega smo sistem razvijali tudi z namenom pomoči tekmovalcem CERN-ovega izziva TrackML, ki je bil razpisan v letu 2018 v CERN-u in objavljen na spletišču Kaggle. Za izboljšave sistema smo dobili pozitiven odziv članov vizualizacijske skupine fundacije HSF. Prav tako smo dobili pozitiven odziv udeležencev TrackML izziva, ki so sistem lahko uporabili kot pomoč in podporo pri reševanju omenjenega izziva.

V delu smo predstavili izhodiščni in izboljšani sistem. Predstavili smo omejitve izhodiščnega sistema in pristope reševanja teh omejitev v izboljšanem sistemu. Opisali smo sestavne dele izboljšanega sistema, ki zajemajo pripravo podatkov, njihovo optimizacijo, senčenje, upodobitev v sceni in uporabniški vmesnik.

Z optimizacijo smo zagotovili visoko stopnjo odzivnosti vizualizacije, kjer smo se osredotočili na minimizacijo klicev za izris. Z uporabo WebGL-a smo zagotovili platformno neodvisnost in z uspešno optimizacijo možnost učinkovite uporabe na vsakdanjih sistemih. Z izbiro pravih parametrov vi-

zualizacije in primerno realizacijo prosojnosti objektov smo uspeli realizirati intuitiven in nazoren prikaz vizualizacije.

Predstavili smo tudi performančno analizo upodabljanja izhodiščnega in izboljšanega sistema, s pomočjo katere smo ju ovrednotili in dobili vpogled v nadaljnji razvoj.

4.1 Ugotovitve in nadaljnje delo

Primerna analiza in ovrednotenje delovanja sistema nam omogoči, da izberemo pravi pristop, ko skušamo sistem izboljšati. Posebno pazljivi moramo biti, ko uporabljamo tehnologije in programske jezike, ki abstrahirajo delovanje in velikokrat zakrijejo delovanje na nižjih nivojih. V poglavju analize in ovrednotenja smo ugotovili, da pri optimizaciji vizualizacije velike količine podatkov zagotovo ne smemo izpustiti razmisleka o delovanju nižjenivojskih struktur, kot so klici za izris. Ugotovili smo tudi, da tehnike kot so indeksriranje, lahko v veliko primerih doprinesejo k dodatni optimizaciji sistema.

V nadaljnjem delu bi sistem zagotovo lahko še dodatno izboljšali in nadgradili. Sistem lahko poskusimo še dodatno optimizirati z minimizacijo uporabe uniform v senčilnikih. Prav tako lahko poskušamo še združiti nekatere atribute geometrije in s tem zmanjšamo prenose v pomnilnikih. S tem izboljšamo tudi poravnave v pomnilnikih, kar v splošnem privede do učinkovitejšega dostopa do podatkov. Izboljšali bi lahko tudi učinkovitost predstavitve prosojnih objektov, predvsem geometrije detektorja. Na tem mestu lahko geometrijo združimo v en objekt, ki ga lahko izrisujemo le z enim klicem za izris, moramo pa poskrbeti za pravilen vrstni red izrisa.

V sistem bi lahko dodali še dodatne funkcionalnosti vizualizacije. Za boljši prikaz posameznih trajektorij bi lahko implementirali možnost sledenja obliki trajektoriji. Vizualizacijo bi lahko tudi obogatili s prikazom dodatnih podatkov, ki si zajeti ob eksperimentu trka delcev. Vizualizacijsko najbolj zanimiva so tu mesta trkov, kjer pride do interakcije delcev z detektorji. V nadaljnjem delu bi lahko tudi nadgradili izračun debeline izbranih trajektorij,

kjer lahko izboljšamo izgled med pregibi posameznih segmentov.

Z uporabo ogrodja Med3D se nam odpira tudi možnost upodabljanja v brskalniku združenega z upodobljevalnim sistemom na namenskem strežniku. To nam omogoča močno fleksibilnost na področju interaktivnosti in oddaljenem sodelovanju. Oddaljeno sodelovanje bi omogočilo uporabo takega vizualizacijskega orodja na področju fizike osnovnih delcev v realnem času na mednarodni ravni. S tako inovativno rešitvijo bi v znanosti fizike spodbudili sodelovanje in omogočili izboljšan napredek v raziskovanju.

Literatura

[1] J. Allison, M. Asai, G. Barrand, M. Donszelmann, K. Minamimoto, J. Perl, S. Tanaka, E. Tcherniaev, and J. Tinslay. The Geant4 Visualisation System. *Computer Physics Communications*, 178(5):331 – 365, 2008.

[2] Matthew Bellis, Riccardo Maria Bianchi, Sebastien Binet, Ciril Bohak, Benjamin Couturier, Hadrien Grasland, Oliver Gutsche, Sergey Linev, Alex Martyniuk, Thomas McCauley, Edward Moyse, Alja Mrak Tadel, Mark Neubauer, Jeremi Niedziela, Leo Piilonen, Jim Pivarski, Martin Ritter, Tai Sakuma, Matevz Tadel, Barthélémy von Haller, Ilija Vukotic, and Ben Waugh. HEP Software Foundation Community White Paper Working Group – Visualization, 2018.

[3] C. Bohak, A. Sodja, M. Marolt, U. Mitrović, and F. Pernuš. Fast segmentation, conversion and rendering of volumetric data using gpu. In *IWSSIP 2014 Proceedings*, pages 239–242, May 2014.

[4] J Boudreau and V Tsulaia. The GeoModel Toolkit for Detector Description. 2005.

[5] Rene Brun and Fons Rademakers. ROOT — An object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1):81 – 86, 1997. New Computing Techniques in Physics Research V.

[6] L Chevalier, J Ernwein, A Formica, P-F Giraud, J-F Laporte, A Ouraou, P Sizun, and M Virchaux. PERSINT Event Display for ATLAS. Technical Report ATL-SOFT-PUB-2012-001, CERN, Geneva, Dec 2012.

[7] LHC Experiments Committee. *ATLAS inner detector: Technical Design Report, 1.* Technical Design Report ATLAS. CERN, Geneva, 1997.

[8] Brian Danchilla. *Three.js Framework*, pages 173–203. Apress, Berkeley, CA, 2012.

[9] Dean Jackson and Jeff Gilbert. WebGL Specification. Technical report, The Khronos Group Inc., 2015.

[10] Khronos Group. *The OpenCL Specification*, September 2010.

[11] Thomas Kittelmann, Vakhtang Tsulaia, Joseph Boudreau, and Edward Moyse. The Virtual Point 1 event display for the ATLAS experiment. *Journal of Physics: Conference Series*, 219(3):032012, 2010.

[12] Primož Lavrič. Spletno ogrodje za vizualizacijo volumetričnih podatkov z možnostjo oddaljenega sodelovanja. Diplomska naloga, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2016.

[13] Primož Lavrič, Ciril Bohak, and Matija Marolt. Spletno vizualizacijsko ogrodje z možnostjo oddaljenega sodelovanja. In *Zbornik petindvajsete mednarodne Elektrotehniške in računalniške konference ERK 2016, 19. - 21. september 2016, Portorož, Slovenija*, pages 43–46, 2016.

[14] Žiga Lesar. Vizualizacija medicinskih volumetričnih podatkov v realnem času. Diplomska naloga, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2014.

[15] T McCauley. A browser-based event display for the CMS Experiment at the LHC using WebGL. *Journal of Physics: Conference Series*, 898(7):072030, 2017.

[16] Rok Oblak. Simulacija in vizualizacija pretoka krvi v ožilju na spletni platformi. Magistrsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2017.

[17] Gabriele Sabato and Markus Elsing. ATLAS Fast Physics Monitoring: TADA. Technical Report ATL-SOFT-PROC-2017-038. 9, CERN, Geneva, Feb 2017.

[18] Tai Sakuma and Thomas McCauley. Detector and Event Visualization with SketchUp at the CMS Experiment. *Journal of Physics: Conference Series*, 513(2):022032, 2014.

[19] Mark Segal and Kurt Akeley. The OpenGL Graphics System: A Specification (Version 3.1). Technical report, The Khronos Group Inc., March 2009.

[20] Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1993.